ABSTRACT

SECURE ENTITY AUTHENTICATION

by
Zuochao Dou

According to Wikipedia, authentication is the act of confirming the truth of an attribute of a single piece of a datum claimed true by an entity. Specifically, entity authentication is the process by which an agent in a distributed system gains confidence in the identity of a communicating partner (Bellare et al.). Legacy password authentication is still the most popular one, however, it suffers from many limitations, such as hacking through social engineering techniques, dictionary attack or database leak. To address the security concerns in legacy password-based authentication, many new authentication factors are introduced, such as PINs (Personal Identification Numbers) delivered through out-of-band channels, human biometrics and hardware tokens. However, each of these authentication factors has its own inherent weaknesses and security limitations. For example, phishing is still effective even when using out-of-band-channels to deliver PINs (Personal Identification Numbers). In this dissertation, three types of secure entity authentication schemes are developed to alleviate the weaknesses and limitations of existing authentication mechanisms: (1) End user authentication scheme based on Network Round-Trip Time ($NRTT$) to complement location based authentication mechanisms; (2) Apache Hadoop authentication mechanism based on Trusted Platform Module (TPM) technology; and (3) Web server authentication mechanism for phishing detection with a new detection factor $NRTT$. In the first work, a new authentication factor based on $NRTT$ is presented. Two research challenges (i.e., the secure measurement of $NRTT$ and the network instabilities) are addressed to show that $NRTT$ can be used to uniquely and securely identify login locations and hence can support location-based web authentication mechanisms. The experiments and analysis show that $NRTT$ has superior usability, deploy-ability, security, and performance properties compared to the state-of-the-art web authentication factors. In the second work, departing from the Kerberos-centric approach, an authentication framework for Hadoop that utilizes Trusted Platform Module (TPM) technology is proposed. It is proven that pushing the security down to the hardware level in conjunction with software techniques provides better protection over software only solutions. The proposed approach provides significant security guarantees against insider threats, which manipulate the execution environment without the consent of legitimate clients. Extensive experiments are conducted to validate the performance and the security properties of the proposed approach. Moreover, the correctness and the security guarantees are formally proved via Burrows-Abadi-Needham (BAN) logic. In the third work, together with a phishing victim identification algorithm, $NRTT$ is used as a new phishing detection feature to improve the detection accuracy of existing phishing detection approaches. The state-of-art phishing detection methods fall into two categories: heuristics and blacklist. The experiments show that the combination of $NRTT$ with existing heuristics can improve the overall detection accuracy while maintaining a low false positive rate. In the future, to develop a more robust and efficient phishing detection scheme, it is paramount for phishing detection approaches to carefully select the features that strike the right balance between detection accuracy and robustness in the face of potential manipulations. In addition, leveraging Deep Learning (DL) algorithms to improve the performance of phishing detection schemes could be a viable alternative to traditional machine learning algorithms (e.g., SVM, LR), especially when handling complex and large scale datasets.

SECURE ENTITY AUTHENTICATION

by
Zuochao Dou

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

May 2018

APPROVAL PAGE

SECURE ENTITY AUTHENTICATION

Zuochao Dou

Dr. Abdallah Khreishah, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Issa Khalil, Committee Member — Date
Principal Scientist of Qatar Computing Research Institute, Hamad bin Khalifa University

---

Dr. Nirwan Ansari, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Roberto Rojas-Cessa, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Reza Curtmola, Committee Member — Date
Associate Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Zuochao Dou
Degree:	Doctor of Philosophy
Date:	May 2018

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2018
- Master of Science in Electrical Engineering,
  University of Rochester, Rochester, NY, 2013
- Master of Science in Computer Science,
  University of Southern Denmark, Sønderborg, Denmark, 2011
- Bachelor of Science in Electrical Engineering,
  Beijing University of Technology, Beijing, China, 2009

Major: Computer Engineering

Presentations and Publications:

Zuochao Dou, Issa Khalil, Abdallah Khreishah, Ala Al-Fuqaha, and Mohsen Guizani, "Systemization of Knowledge (SoK): A systematic Review of Software Based Phishing Website Detection Research," *IEEE Communications Surveys & Tutorials* , vol. 19, issue. 2, pp. 2797-2819, fourth quarter 2017.

Zuochao Dou, Issa Khalil, and Abdallah Khreishah, "A Novel and Robust Authentication Factor Based on Network Communications Latency," *IEEE Systems Journal*, DOI: 10.1109/JSYST.2017.2691550, 2017.

Zuochao Dou, Issa Khalil, Abdallah Khreishah, and Ala Al-Fuqaha, "Robust Insider Attacks Countermeasure for Hadoop: Design & Implementation," *IEEE Systems Journal*, DOI: 10.1109/JSYST.2017.2669908, 2017.

Zuochao Dou, Issa Khalil, and Abdallah Khreishah, "CLAS: A Novel Communications Latency based Authentication Scheme," *Security and Communication Networks*, 2017.

Issa Khalil, Zuochao Dou, and Abdallah Khreishah, "Your Credentials Are Compromised, Do Not Panic: You Can Be Well Protected," *ACM Asia conference on Computer & Communications Security (AsiaCCS)*, 2016.

Issa Khalil, Zuochao Dou, and Abdallah Khreishah, "TPM-based Authentication Mechanism for Apache Hadoop," *10th International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2014.

Kwon Minseok, Zuochao Dou, Wendi Heinzelman, Tolga Soyata, He Ba, and Jiye Shi, "Use of Network Latency Profiling and Redundancy for Cloud Server Selection," *IEEE 7th International Conference in Cloud Computing (CLOUD)*, 2014.

Hazim Shakhatreh, Ahmad Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Othman, Abdallah Khreishah, and Mohsen Guizani, "Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges," *IEEE Communications Surveys & Tutorials*, under review, 2018.

To my parents, ChangXing Dou and XiuRong Yang; my sister, RuiJun Dou; my brother-in-law, Xing Cao; my wife, Yifan Wang who have always loved me, believed in me, encouraged me, and supported me through everything.

勇敢的少年，快去创造奇迹！

ACKNOWLEDGMENT

First of all, I would like to express my deepest appreciation to my advisor and co-advisor, Dr. Abdallah Khreishah and Dr. Issa Khalil for their guidance and patience in the last four years. It is been a pleasure working under such two great professors.

In addition, I would also like to express my gratitude to Dr. Nirwan Ansari, Dr. Roberto Rojas-Cessa and Dr. Reza Curtmola for honoring me as members of my dissertation committee. I also thank them for their feedback on this research.

I also want to thank all the professors who I have collaborated with. I would especially like to thank Dr. Ala Al-Fuqaha, and Dr. Mohsen Guizani for the useful discussions I had with them. These discussions opened my eyes on some important things both in life and this research.

Finally, I thank my parents, my sister, and my wife for their support and faith in me.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

Authentication is the process of gaining assurance that an entity is performing robustly and precisely as intended [132] [33]. The authentication entity could be an end user, a web server, a service or even a hardware platform. The authentication credentials for a platform/service could be a certificate (e.g., X.509 certificate [59]), a ticket (e.g., Kerberos ticket [99]), or a hardware signature (e.g., Physical Unclonable Functions [113]) etc. On the other hand, the authentication credentials for a user fall into three categories: (1) Something you know (e.g., a password) which is the most common kind of authentication used for humans. (2) Something you have (e.g., a smart card) that some object must be with the user any time he want to be authenticated. (3) Something you are (e.g., a fingerprint) which is based on something intrinsic to the principal being authenticated.

In this dissertation, we present three types of entity authentication works using existing and novel techniques: (1) End user authentication: a novel and robust authentication factor based on network communications latency (Chapter 2); (2) Web server authentication: a novel and robust phishing detection feature (Chapter 3); and (3) Platform and service authentication: robust insider attacks countermeasure for Hadoop: design & implementation (Chapter 4); The dissertation is finally summarized in Chapter 5.

1.1 End User Authentication

The proliferation of cloud-based services (Gmail, Dropbox, Amazon, Facebook, etc.) and other web services (social sites, E-commerce, etc.), makes end user authentication a very important security mandate to enable secure interaction with such services. Each of the traditional web authentication approaches (e.g., legacy password, multi-factor authentication, etc.) has its inherent weaknesses and suffers from one or more of the following limitations: (1) potential compromise of credentials through, for example, internal observation, social engineering, spyware, and leakage from other verifiers; (2) having single point of failure as in the case of Facebook Connect [93]; (3) vulnerable to active man-in-the-middle attacks through phishing or pharming [47]; and (4) sometimes having poor user experience due to typing of extra bits of information or using extra channel or device.

Legacy password authentication is still the most popular one, however it suffers from many obvious usability and security limitations. For example, users who have multiple web accounts have either to memorize multiple passwords (poor usability) or use the same password on multiple accounts (poor security). There have been many attempts to enhance usability of legacy passwords including LastPass [10], Facebook Connect [7], and federated passwords such as OpenID [107]. LastPass remembers user password and fills in the corresponding password fields automatically, when required. Facebook Connect enables users to sign on into different web services using their same (e.g., Facebook) authentication credentials. Similarly, OpenID [107] simplifies web-access by maintaining all the passwords of a user for different website, and requiring the user to only remember one password, the OpenID password. Then, while browsing, OpenID provider presents the appropriate password of the user to other websites, that is, OpenID provider authenticates on behalf of the user. While such attempts enhance usability, it badly hurts security because attackers gain the benefit of compromising many web services with single password compromise. To enhance the security of legacy passwords, graphical passwords [121] and cognitive passwords [130] have been introduced. However, the proliferation of web services combined with legacy password vulnerabilities fuels authentication based attacks as evidenced by RSA study in 2015 [5], which shows that 80% of successful cyber attacks exploit authentication credentials.

To overcome the inherited weaknesses of legacy password mechanisms, 2-factor authentication (2FA) schemes were introduced [108]. 2FA schemes require not only a password but also an additional piece of information shared only with the user via out-of-band channels or devices, such as phone SMS and hardware tokens (e.g., RSA SecurID [13]). However, 2FA schemes introduce new vulnerabilities and suffer from the following limitations:

1. 2FA does not protect against man-in-the-middle attacks (through, e.g., phishing [62, 65]).
2. Additional information is exchanged with user through different channels/devices that are likely to be compromised. For example, smartphone SMS is among the most widely used 2FA channels but the smartphone itself is vulnerable to loss and theft (e.g., 4.7 million phones were lost or stolen during 2013 in USA only [4]). Moreover, smartphones are becoming more and more susceptible to mobile malware/spyware infections. For example, earlier this year, Symantec revealed an active Android malware that can intercept SMS messages with 2FA codes and forwards them to attackers [16]).
3. 2FA may have (i) poor usability, due to typing of extra bits of information and (ii) poor accessibility, due to the use of extra channel/device.

Biometric-based approaches have also been considered to support secure authentication by leveraging the uniqueness of physical or behavioral characteristics of individuals [129]. Nevertheless, there are quite a few limitations that prevent it from being widely adopted as a web authentication mechanism:

1. Similar to 2FA, biometric authentication schemes have no protection against active man-in-the-middle attacks for web access.
2. Biometric schemes have even worst user experience due to the extra overhead required to characterise individuals, e.g. scanning or recording user's physical or behavioral characteristics.
3. Often a special device is required for biometric input, which incurs extra cost.

In order to complement the state-of-the-art web authentication schemes by alleviating many of their inherent weaknesses and vulnerabilities, we propose a new authentication factor based on Network Round Trip Time ($NRTT$). We show how $NRTT$ can be used to uniquely and securely identify login locations and hence can support location-based web authentication mechanisms. The first research challenge is how to securely measure and verify $NRTT$ to hamper potential forgery attempts. We address the first challenge by introducing a novel forwarding device in the path between the server and the client, dubbed delay mask (DM), which prevents any entity, but the server, from being able to measure the $NRTT$ for any client. The second research challenge is how to reliably measure $NRTT$ in the face of variable Internet latencies and connectivity conditions. The second challenge is addressed by (1) computing the average of a number of $NRTT$ measurements after outlier removal; (2) applying multiple profiles per user through the deployment of multiple DMs in diverse geographical locations. We design a two-factor authentication scheme (dubbed AMAN) that uses legacy passwords as a first factor and $NRTT$ as a second authentication factor. We conduct extensive experiments to evaluate Security-Usability-Deployability properties of AMAN and compare it with state-of-the-art authentication mechanisms. The results show that AMAN achieves the best combination of these properties.

1.2 Web Server Authentication

Phishing has been defined in various ways. According to PhishTank [100]: "Phishing is a fraudulent attempt, usually made through email, to steal personal information". This definition covers most of the cases in which phishing attackers aim at stealing personal information such as authentication credentials. In a classic email phishing scenario, an attacker hosts a fake website and presents web service users with convincing emails containing a link to the fake website. When any web service user opens the link and enters his sensitive data, the data will be collected by the server hosting the fake website.

Khonji et al. [75] defines phishing as a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit. This definition limits the phishing tactic to only social engineering approaches.

In this dissertation, we adopt the definition provided in [133]: "We define a phishing page as any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party". This definition neither limits the attacker's goal (e.g., to steal personal information) nor limits the attack strategy (e.g., through social engineering messages or sophisticated techniques).

According to the 2015 phishing activity trends report [51], the total number of unique phishing sites detected from January through September was 630,494. In addition, the number of phishing websites increased 250% from the 4th quarter of 2015 through the 1st quarter of 2016 [1]. There is no agreement on the financial damage caused by the phishing attacks due to the lack of data from victim institutions. Some estimates show that the direct damage caused by phishing attacks ranges from \$61 million per year to \$3 billion per year within the U.S. alone [58]

Phishing attacks tend to use more sophisticated techniques to lure web service users into a carefully designed rogue website. On one hand, phishing attackers become more careful and attentive in designing phishing websites and attempt to evade current phishing detection methods[1].

More importantly, some phishing groups have the ability and desire to perform advanced phishing attacks. Avalanche (commonly known as the Avalanche Gang) is a criminal syndicate involved in phishing attacks [135]. In 2010, the Anti-Phishing Working Group (APWG) reported that Avalanche had been responsible for two-thirds of all phishing attacks in the second half of 2009, describing it as "one of the most sophisticated and damaging on the Internet" and "the world's most prolific phishing gang" [18]. It has been proved that Avalanche uses different techniques to evade the anti-phishing mechanisms.

Anti-Phishing techniques can be broadly classified in two different categories: (1) Server side solutions such as brand monitoring[20]; and (2) Client side solutions such as blacklists and heuristics techniques[58]. Alternatively, anti-Phishing techniques can be classified into 3 categories: (1) preventive solutions such as anti-malware, (2) detective solutions such as email filtering; and (3) corrective solutions such as Site takedown.

In this work, we focus on the area of phishing detection. Our goal is to detect and block Phishing websites immediately after the user clicks its malicious link. This area is of great importance because if a person behind the keyboard has been successfully fooled by the Phishing attempt, it doesn't help how many firewalls, encryption software, certificates, or two-factor authentication mechanisms an organization provides [58]. Therefore, we focus here in strengthening the last line of defence against Phishing by enhancing the chances to detect Phishing attempts and warning victims before being redirected to the suspicious websites.

The two commonly used ways of client side Phishing detection are heuristics and blacklists [58]. Heuristics methods examine the contents of web pages including: (1) surface level contents such as domain name and URL, (2) textual contents such as words that appear in a given web page, and (3) visual contents such as the layout and the block regions [91]. These techniques can detect Phishing attacks as soon as they are launched. However, they introduce relatively high false positive rate. On the other hand, manually verified blacklist has higher level of accuracy. However, they do not defend against zero-hour attacks.

In this work, we propose a new phishing detection framework by carefully combining different phishing detection features together with the phishing target identification. Based on the most recent phishing trends, we first perform analysis of the state-of-the-art phishing detection features collected from the widely used and popular detection mechanisms. In addition, we introduce a novel phishing detection factor - Network Round Trip Communications Time ($NRTT$) with corresponding analysis. Then, we present a novel algorithm to identify the target website from suspicious URL, which is not well addressed by current literature. The evaluation of the proposed framework show that our proposed mechanism not only has better performance results but also can neutralize advanced phishing attacks.

1.3 Platform Authentication

Authentication is the process of gaining assurance that an entity is performing robustly and precisely as intended [132] [33]. In addition, data confidentiality in the cloud is tightly correlated to the user authentication [140]. Therefore, a secure and robust authentication mechanism of both users and services is imperative for secure and private cloud computing and storage operations [86]. However, the continuous growth and the concentration of data in clouds, combined with the increasing adoption of security solutions such as authentication, access control, and encryption drives intruders to be more persistent and creative in developing sophisticated attack strategies [122]. One way to protect clouds and to successfully combat such sophisticated attacks is to push the bar higher through the combination of hardware and software security solutions. Pushing the security down to the hardware level in conjunction with software techniques provides better protection over software-only solutions [104], which is especially feasible and suitable for entity authentication and platform attestation in the cloud.

Hadoop provides an open source framework for the storage and parallel processing of large-scale data sets on clusters of commodity computers. As the amount of data maintained by industrial corporations grows over time, big data processing becomes more important to enterprise data centers.Hadoop and its MapReduce programming model have been proposed to address this task. This model is used to handle large-scale data sets with high efficiency by taking advantage of parallel data processing.

However, the threat of data leaks also continues to grow due to the increasing number of entities involved in running and maintaining cloud infrastructure and operations [140]. The recent boost of big data start-ups such as MongoDB, DataStax, MapR Technologies and Skytree leads to an increased number of points of access in the cloud, that is, larger attack surface for intruders. This can be clearly inferred from a recent report of the Health Information Trust Alliance (HITRUST), which reveals that the total cost of health-care data breach incidents has grown to $4.1 billion over the recent years [56].

Currently, Hadoop leverages Kerberos [114] [21] as the primary authentication method and uses DIGEST-MD5 security tokens [80] to supplement the primary Kerberos authentication process. This Kerberos based authentication mechanism was initially implemented by a team at Yahoo in 2009 [80]. However, in addition to its limitations and security weaknesses, the use of Kerberos for authentication in Hadoop-based environments raises many security concerns.

The most vital weakness of Kerberos lies in its dependency on passwords. The session key for data encryption during the initial communication phase with the Key Distribution Center (KDC) is derived from the user's password. Disclosure of KDC passwords allows attackers to capture users' credentials, which turns all Hadoop's security to be useless. The large number of password disclosure incidents through cracking, social engineering, or even database leakage, clearly indicates that this threat is real and pervasive.

It has been shown that in many situations passwords are relatively easy to break (e.g., via hardware key-loggers, spear Phishing with malware, shoulder surfing, etc.)

For example, in 2013, almost 150 million people have been affected by a breach into Adobe's database [37]. The breach is due to mistakes made by Adobe in handling clients' passwords. All passwords in the affected database were encrypted with the same key. Additionally, the encryption algorithm used did not handle identical plain texts properly. This resulted in similar passwords being encrypted into similar ciphers.

Another important issue of Kerberos lies in its dependency on the KDC which constitutes a single point of failure and even a single point of attack for persistent and dedicated attackers. Although Hadoop's security design introduces delegation tokens to overcome this bottleneck, they lead to a more complex authentication mechanism due to the extra tokens and data flows that are required to enable access to Hadoop services. Many types of token have been introduced, including delegation tokens, block tokens, and job tokens for different subsequent authentications. This, relatively, large number of tokens, not only complicates the configuration and the management of the tokens, but also expands the attack surface [27]. Kerberos keys are stored in an on-line third-party database. If anyone other than the proper user has access to the KDC, through, for example, a malware installation by an insider, the entire Kerberos authentication infrastructure will be compromised and the attacker will be able to impersonate any user [54]. This highlights the fact that insiders could create havoc in Kerberos infrastructure itself, and consequently affect the security posture of the supported Hadoop. It is clear that Kerberos is not well-equipped against insiders or outsiders who could change the execution environment that the user trusts. For example, attackers may install key loggers or other malware-tools to steal users' credentials and data.

In this work, we depart from the Kerberos-based approach to propose a TPM-based authentication protocol for Hadoop. To date, more than 500 million PCs have been shipped with TPMs, an embedded crypto capability that supports user, application, and machine authentication with a single solution [104]. Additionally, many virtual TPM implementations exist for virtualized environments [32] [110]. An application that can be developed using the software TPM will run using a hardware TPM without changes [88]. TPM offers facilities for the secure generation of cryptographic keys, and limitation of their use, in addition to a random number generator.

Beyond providing the regular authentication services supported by Hadoop, our protocol provides additional security services that are not provided by the current state-of-the-art Hadoop authentication protocols. In addition to alleviating the aforementioned security weaknesses of Kerberos, our protocol guards against any tamper with the hardware or software of the target cloud machines that store and process users' encrypted data. The user cannot be presumed to trust the execution environment on public clouds. Malicious insiders/outsiders can pose great threats to users' data even if it is encrypted in the steady state. Insiders may be able to install malicious software (malware, spyware, etc.) and hardware (key loggers, side channels, etc.) tools that can extract users' data and sensitive credentials while it is being processed. The data has to be decrypted before any processing can be performed.

CHAPTER 2

A NOVEL AND ROBUST AUTHENTICATION FACTOR BASED ON NETWORK COMMUNICATIONS LATENCY

2.1 Introduction

Legacy password authentication suffers from many obvious usability and security limitations. The credentials of the users not only hacked through social engineering and dictionary attacks, but also databases storing such credentials have been hacked, exposing massive number of user accounts [64] [74]. To address the security concerns in legacy password-based authentication, many new authentication factors have been introduced and tested, including: (1) random strings delivered through out-of-band channels such as mobiles and emails; (2) human biometrics such as fingerprints and iris scans; (3) profile-based factors such as profiling normal user behavior, browser fingerprinting, IP address information, and login location; (4) physical factors such as cards, hardware tokens, and mobiles; (5) knowledge-based factors such as recognizing someone based on photos provided by social websites [30]. However, each of these authentication factors has its own inherent weaknesses and security limitations. For example, phishing is still effective even when using out-of-band-channels to deliver second factor Personal Identification Numbers (PINs) or passwords. Internal observation can also defeat many of these factors, especially human biometrics and system fingerprinting [139]. Additionally, some of these authentication factors are static, such as browser fingerprints and IPs, and hence can be forged or leaked across different verifiers. Physical factors, on the other hand, can be lost, stolen, or compromised. Furthermore, some factors have high false negative rate such as keyboard typing rhythms and end user profiling. It is also worth noting that, in addition to the security limitations, many of these factors have usability issues due to the requirement of extra information, devices, or channels. Appendix 2.6 provides a more detailed and systematic analysis and comparisons of the vulnerabilities of the state-of-the-art authentication factors.

In this work, we propose a new authentication factor that does not share the above mentioned properties with the commonly used authentication factors. That is, the new authentication factor is oblivious to clients and is not communicated to the server, but rather is completely measured and verified at the server. Our proposed authentication factor utilizes what initially appears to be counter-intuitive, the Network Round-Trip communications Time ($NRTT$). $NRTT$ is defined as the summation of the time a packet takes to travel from the server to the client and the time its acknowledgment takes to travel back from the client to the server.

In this work, we show how to turn the insecure and potentially unstable $NRTT$ into a robust authentication factor that is resilient to both client compromise and communication channel compromise. $NRTT$ offers unique security features such as resiliency to Phishing, MitM, leakage by other verifiers, and social engineering, which complements the security features of other authentication factors. Moreover, $NRTT$ has the advantage of being user transparent (i.e., it does not require clients to memorize or input any information) and has negligible overhead, which enables it to be smoothly integrated with other authentication factors in multi-factor authentication schemes without introducing extra overhead or degrading usability. However, $NRTT$ can only be used to provide authentication for low mobility users and static users, similar to location-based authentication schemes. It is intuitive to see that arbitrarily mobile users cannot benefit from authentication based on $NRTT$ because such users require to be able to login from any arbitrary location, while $NRTT$ can only accept logins from previously profiled locations. Nevertheless, $NRTT$ provides reliable and secure location-based authentication, which is generally used to ensure that users can perform sensitive operations (e.g. change password, initiate funds transfers) or access valuable information (e.g., personal medical information) only from authorized locations. Additionally, secure location identification is important for other security purposes. For example, geographical location is one of the most commonly used indicators to detect phishing based on the observation that phishing websites are most likely to be hosted in locations different from those of the corresponding legitimate websites [2].

We summarize our contributions in this work as follows:

- Propose a novel secure and usable web authentication factor based on Network Round Trip Time, $NRTT$.
- Provide comparative evaluation of $NRTT$ against state-of-the-art authentication factors using a famous authentication benchmark framework.
- Design and implement a novel network architecture that enables secure measurement of $NRTT$.
- Design and implement algorithms to alleviate network instabilities and expand authentication sample space of $NRTT$.
- Design, implement and deploy a prototype for a use case of two-factor authentication (AMAN) with legacy passwords as the first factor and $NRTT$ as the second factor. The prototype helps to practically evaluate the security, usability, and deploy-ability properties of $NRTT$-based two factor authentication and to assess its performance overhead.
- Provide mathematical analysis of the $NRTT$ space space via a case study.

2.2 General Methodology to Use $NRTT$ As an Authentication Factor

Authentication using $NRTT$ is straightforward. At registration, $NRTT$ statistics (i.e., mean and standard deviation) between the client and the authenticator are measured and stored at the authenticator as a reference profile. The $NRTT$ statistics are then re-measured with every login attempt in real-time, and login is granted only if the new statistics fall within predefined boundaries from the corresponding registration statistics.

It has been observed, through extensive experiments and monitoring of Internet communications, that $NRTT$ follows distributions that can be modeled as an approximate Gaussian distribution ([80] [120] [125]) as detailed in Section 2.2.1. We adopt Gaussian approximation to theoretically guide the selection of different $NRTT$ related parameters in our experiments and theoretical analysis. The mean and the standard deviation of $NRTT$ measurements vary when the login location changes, that is, users can be uniquely identified based on their login locations. This important observation indicates that a login attempt will only succeed if conducted from the same location as that of registration, which reduces the attack surface of compromised identities from anywhere in the world to only the registration location. Note that, location-based authentication is very important in many areas, such as electronic health record access, sensitive financial transactions, military communications, industrial control systems, etc. [52, 25] [25] [97] [40]. In addition, within this work, the login location refers to the last network segment, access point or 3G/4G cell of the communicating party.

2.2.1 Gaussian Approximation of $NRTT$

$NRTT$-based authentication is motivated by the results presented in [80], which show that network communications latency approximately follows a Gaussian distribution. This observation is validated by experiments that measure network communications latency among 130 PlanetLab nodes [37].

We have also conducted extensive and wider set of similar experiments using GENI nodes, campus and residential users both with wire-line and wireless connections. Our results validate the results in [80] and further support the observations about the Gaussian approximation of network communications latency. Figure 2.1 shows examples of $NRTT$ distributions and the corresponding Gaussian approximations for three different locations.

Though fine-grained mathematical model (e.g., Rayleigh distribution) may provide a better approximation, it will introduce much more complicated theoretical

Table 2.1 List of All the Acronyms

μ	Mean of the reference profile
σ	Standard deviation of the reference profile
$1-\alpha_i$	Confidence level
δ	Confidence interval (error tolerance)
x	Mean of the real-time profile
y	Standard deviation of the real-time profile
N	Profiling sample size

analysis with marginal or no additional benefit for the real world implementation of the proposed algorithms.More importantly, the empirical results of many of the existing research on round trip network communications latency ([80] [120] [125]) show that Gaussian distribution is an adequate approximation for $NRTT$. These conclusions are further supported by our experiments and mathematical analysis based on the Gaussian approximation of $NRTT$.

2.2.2 Required Sample Size to Reconstruct the $NRTT$ Profile

$NRTT$ profile is built by exchanging a number of small packets, dubbed profiling signals, with the user. The number of profiling signals is known as the profiling sample size. The larger the number of profiling signals, the more accurate the profile will be. However, the larger the number of profiling signals, the higher the bandwidth overhead and the longer the login latency. Therefore, it is critical to find a profiling sample size that leads to an acceptable trade-off between profile accuracy, bandwidth overhead, and the average time it takes a user to login.

To have an initial estimate of the profiling sample size, we use the Gaussian approximation of $NRTT$ distribution. Assume a population with Gaussian distribution that has standard deviation σ and mean μ. The goal is to find the minimum sample size, N, that produces a mean, x, within a certain error margin (aka, error tolerance), δ, with a certain confidence level, $1 - \alpha$. The error margin δ, is the maximum allowed distance between μ and x. The confidence level represents how confident we are that the measured mean (x) falls within the confidence interval. For Gaussian distributions, it has been shown ([11]) that the minimum sample size N can be calculated as:

$$N \geq (Z_{1-\alpha}/\delta)^2\sigma^2 \tag{2.1}$$

Where Z is the critical value for the normal distribution. In other words, for a sample size of N, we are 1-α confident that the measured mean (x) will fall in the range of:

$$\mu - \delta \leq x \leq \mu + \delta \tag{2.2}$$

Similarly, the range of the real-time measured standard deviation σ can be computed using Chi-Square (χ) table as:

$$\sqrt{\frac{\chi_L^2 \cdot S^2}{N-1}} \leq \sigma \leq \sqrt{\frac{\chi_R^2 \cdot S^2}{N-1}} \tag{2.3}$$

Where χ_L and χ_R are computed for specific values of α using the Chi-Square table.

2.2.3 Mathematical Analysis of the $NRTT$ Sample Space

False negative rate (FN): As mentioned in the previous section, the FN rate of the Gaussian distribution is α, which stands for the probability that a legitimate user fails to authenticate from her profiled location. Let $\delta_i = C_i \cdot S_i$ where C_i is the error tolerance coefficient for user i and let U be the total number of users. To compute FN, we plug δ_i in (2.1):

$$Z_{1-\alpha_i} = C_i \cdot \sqrt{N_i}$$

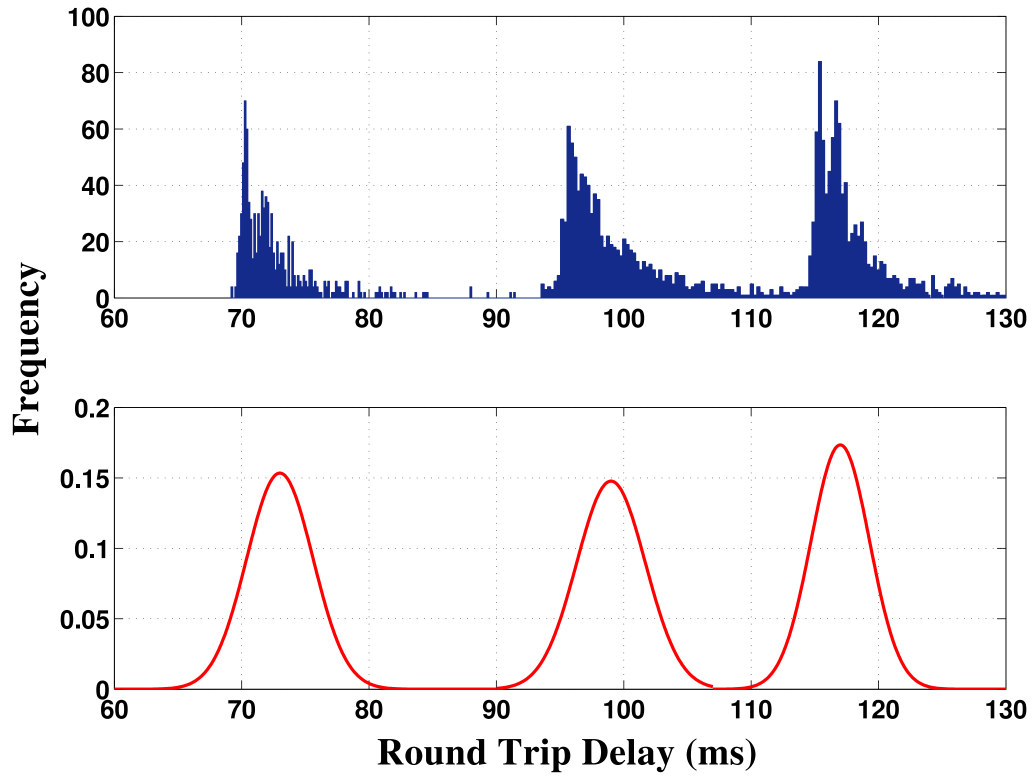


Figure 2.1 Sample of $NRTTs$ and their Gaussian approximations.

$$FN = \sum_{i=1}^{U} (1 - Z^{-1}(C_i \cdot \sqrt{N_i}))/U \tag{2.4}$$

False Positive Rate (FP): The FP, β, is the probability that a perpetrator passes authentication from a location other than the profiled one. In other words, false positive rate is the probability that the real-time measured latency mean and standard deviation of the perpetrator falls within the grant-access area of the legitimate user. We first derive a simplified estimate of the false positive rate and then enhance the derivation accuracy. Figure 2.4 shows the grant-access area for an arbitrary user (User i) and the grant-access area for a perpetrator (Attacker j) at a random location. Recall that the perpetrator also possesses the username and password of the legitimate user. Assume, for now, uniform distribution of the measured mean

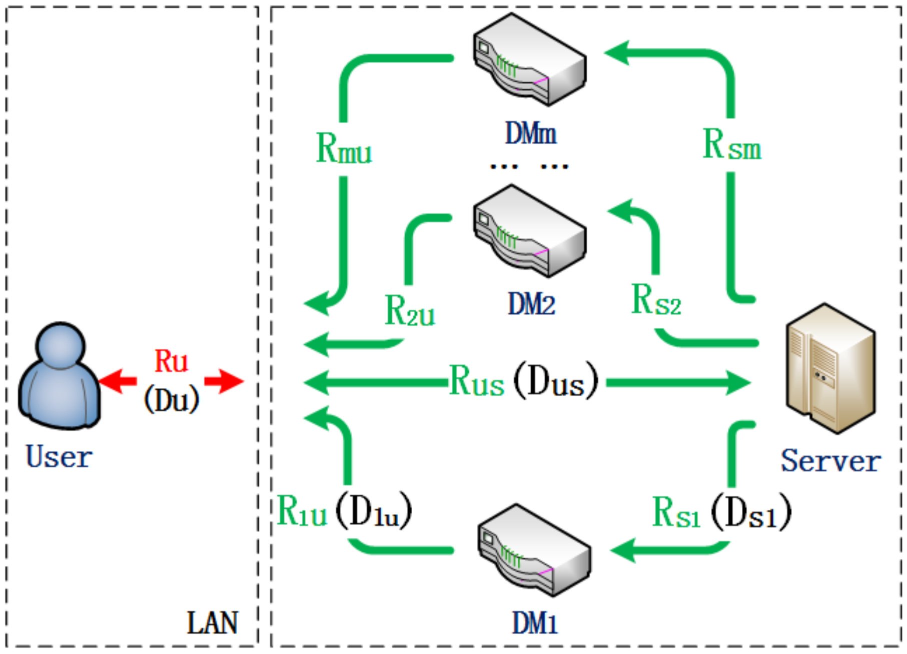


Figure 2.2 The Ecosystem of AMAN.

and standard deviation within the grant-access area (the green rectangle in Figure 2.4 shows the reference profile area of an arbitrary user (User i), dubbed as the grant-access area (GAA)). Access is granted for any login attempt with measured (μ, σ) point that falls within the grant-access area.). Also assume that the locations from which an attacker may try to login are known. Then, the probability that Attacker j successfully authenticates as User i equals the overlap area between the grant-access area of the user and the grant-access area of the perpetrator divided by the grant-access area of perpetrator averaged over all possible attack locations:

$$\beta_i = \sum_{j=1}^{A} \frac{GAA_i \cap GAA_j}{GAA_j} \Big/ A \tag{2.5}$$

Where, A is the number of all possible locations from which the attacker may try to impersonate the user. The overall false positive rate of the system is computed as the average of false positive rates of all the users of the system:

$$FP = \sum_{i=1}^{U} \beta_i / U \tag{2.6}$$

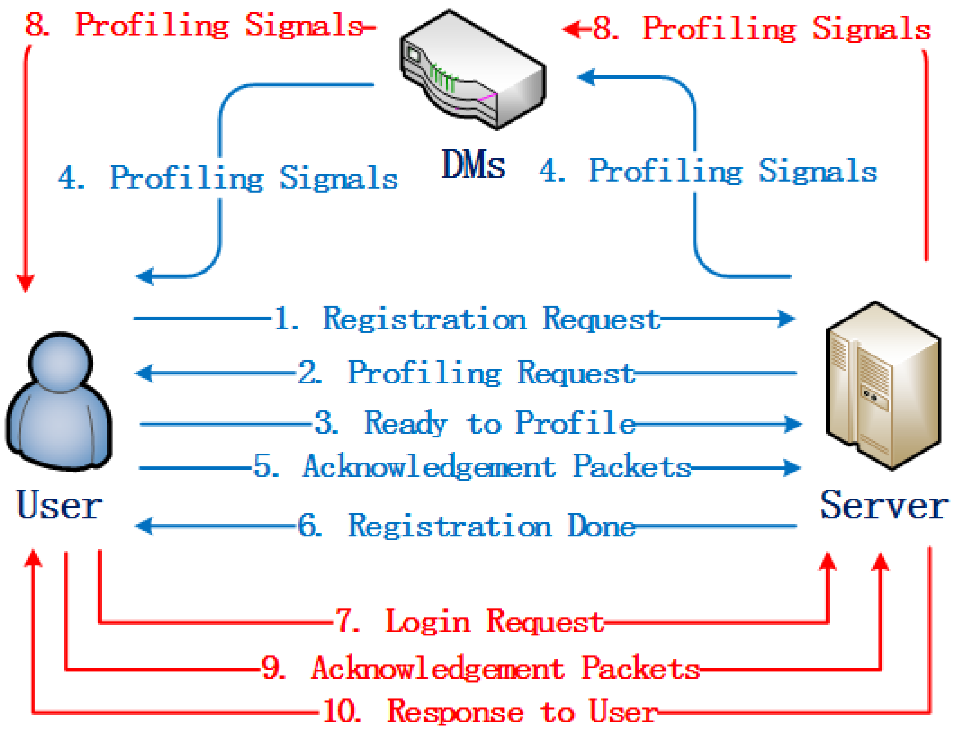


Figure 2.3 AMAN Authentication Flowchart.

Note that even though this is a simplified estimate of the false positive rate, we next show that it provides an upper bound approximation of the false positive rate.

To derive a more accurate estimate of the false positive rate, we need to identify the real distribution of latency mean and standard deviation within the grant-access area, rather than just assuming it to be uniform. Moreover, we need to remove the assumption of previously known attack locations by acknowledging that attackers may use any arbitrary previously unknown location to login. Let the mean of network communications latency be a random variable X in the range $[a, b]$ and let the standard deviation of the mean X be a random variable Y in the range $[c{*}X, d{*}X]$. According to the conclusions derived in [4], which is also validated by our experiments, both X and Y are approximately Gaussian with the following probability distribution functions (pdf):

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \cdot e^{-\frac{(x-\mu_x)}{2\cdot\sigma_x^2}}$$

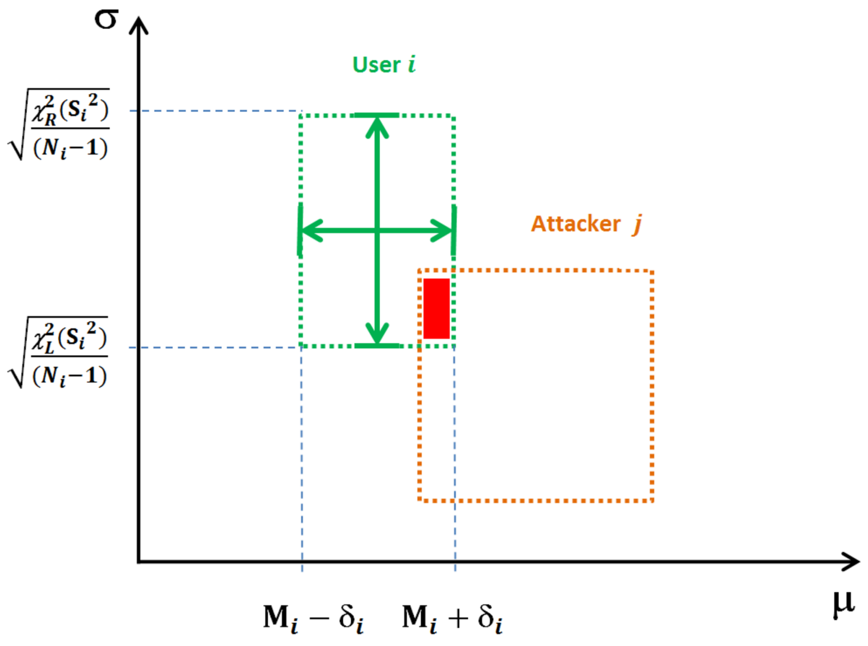

σ
$\sqrt{\frac{\chi_R^2(S_i^2)}{(N_i-1)}}$
User i
Attacker j
$\sqrt{\frac{\chi_L^2(S_i^2)}{(N_i-1)}}$
$M_i - \delta_i$
$M_i + \delta_i$
μ

Figure 2.4 Graphical illustration of arbitrary grant-access area.

$$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \cdot e^{-\frac{(y-\mu_y)}{2\cdot\sigma_y^2}} \tag{2.7}$$

For the systems analyzed in [4], $a = 5ms$, $b = 700ms$, $c = 0.0155$, $d = 0.196$. Therefore, the sample space of the communications latency X and its standard deviation Y falls in the shaded area shown in Figure 2.5. Based on these values, the pdf parameters of the X and Y distributions are:

$$\mu_x = 221, \quad \sigma_x = 83.86 \tag{2.8}$$

$$\mu_y = 0.0155x + (0.196x - 0.0155x)/2 = 0.1058x \tag{2.9}$$

$$\sigma_y = (y_{0.005} - \mu_y)/Q^{-1}(0.005) = 0.035x \tag{2.10}$$

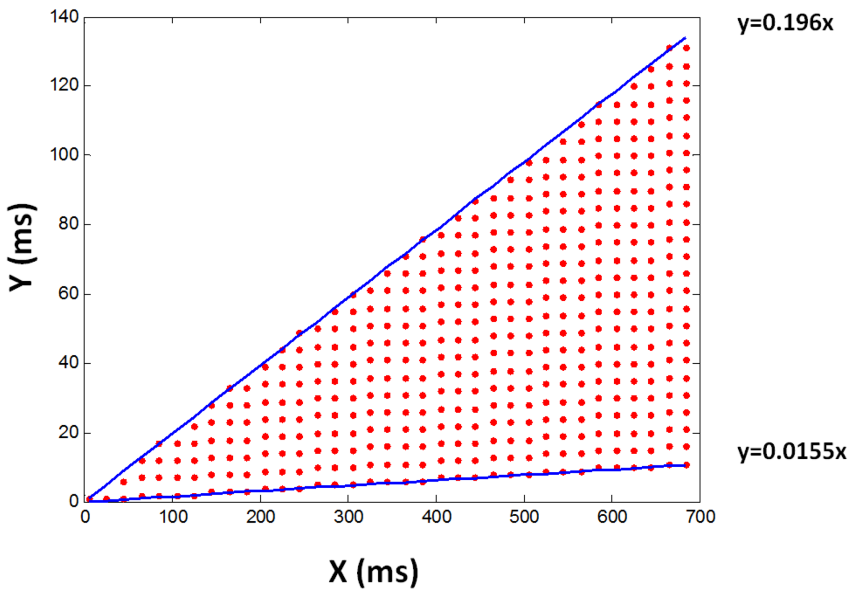


Figure 2.5 Sample space of the communications latency mean and standard deviation.

Using Figure 2.4 and assuming error tolerance $\delta = C \cdot S$, the grant-access area can be computed as:

$$GAA = [(M+\delta) - (M-\delta)] \cdot [\sqrt{\frac{\chi_R^2 \cdot S^2}{N-1}} - \sqrt{\frac{\chi_L^2 \cdot S^2}{N-1}}]$$

According to the experimental results, the optimal false positive rate occurs when $\delta = 0.2 \cdot S$. Therefore,

$$GAA = S^2/10 \tag{2.11}$$

Using (2.7), (2.8), (2.9), (2.10), (2.11), the expected value of the grant-access area is computed as:

$$\int_{x=a}^{b} \int_{y=c\cdot x}^{d\cdot x} \frac{S^2}{10} \cdot \frac{1}{\sqrt{2\pi\sigma_x^2}}$$

$$\cdot e^{-\frac{(x-\mu_x)}{2\cdot\sigma_x^2}} \cdot \frac{1}{\sqrt{2\pi\sigma_y^2}} \cdot e^{-\frac{(y-\mu_y)}{2\cdot\sigma_y^2}} dxdy$$

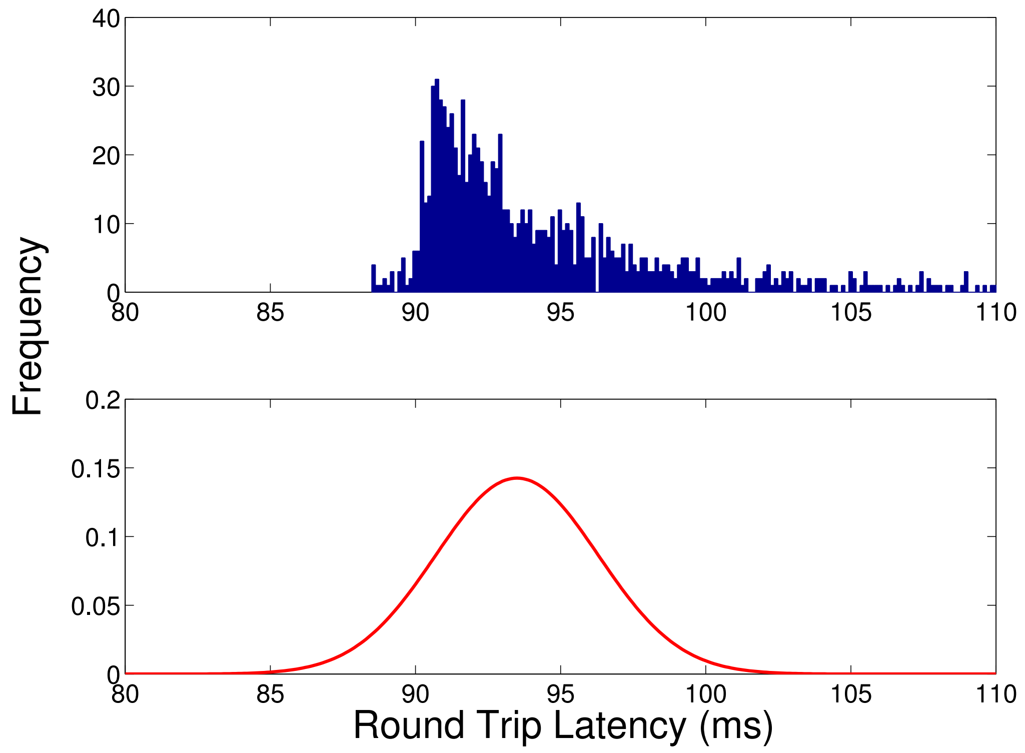


Figure 2.6 Sample of communications latency distribution and its Gaussian distribution.

$$= 4.74 \times 10^{-17} \cdot \int_{x=5}^{700} \int_{y=0.0155x}^{0.196x} \frac{y^2}{x} \cdot e^{-7.11\times10^{-5}\cdot x^2}$$

$$\cdot e^{0.0314x+86.367\cdot\frac{y}{x}-408.16\cdot\frac{y^2}{x^2}} dxdy \tag{2.12}$$

The false positive rate is the probability that an attacker at a random location successfully impersonates a legitimate user, which is given by:

$$FP = \frac{E[GAA]}{Area\ of\ the\ Sample\ Space} \tag{2.13}$$

Using (2.13), for the systems analyzed in [4], the expected value of the false positive rate is approximately 0.0034. In other words, if the attacker tries to login from 1000 different locations, on average, she successfully authenticates from less than 4 of them.

This is a very low probability and hence clearly proves the high security guarantees of AMAN.

Using (2.6), for the systems analyzed in [4], the expected value of the false positive rate is approximately 0.0068. Therefore, the simplified analysis provided earlier provides an upper bound estimate of the false positive rate.

False Positive and False Negative Trade-offs: Optimally, we need to keep both false positive and false negative rates very low. However, these two indicators are dependent. Decreasing the false negative rate increases the false positive rate and vice versa. A possible trade-off is to maximize the security guarantees while maintaining an acceptable functionality level. Using the false positive and false negative formulas developed in Section 2.2.3, the trade-off can be translated into the following optimization problem:

$$\textbf{Minimize } FP = \frac{\sum_{i=1}^{U} \beta_i}{U} = \frac{\sum_{i=1}^{U} \sum_{j=1}^{A} \frac{GAA_i \cap GAA_j}{GAA_j}}{A \cdot U}$$

$$\textbf{S.T. } FN = \sum_{i=1}^{U} [1 - Z^{-1} \cdot (C_i \cdot \sqrt{N_i})]/U \leq FN_{required}$$

$$C_i \geq 0,\ \ N_i,\ U,\ A \geq 0, integer \tag{2.14}$$

Security administrators can use the optimization problem in (2.14) to guide their functionality and security configurations.

2.3 Research Challenges to Use $NRTT$ As an Authentication Factor

The naive measurement and usage of $NRTT$ has two main challenges:

Research Challenge I: Attackers can easily estimate $NRTT$ for any location by simply pinging the server from that location. This represents a serious security challenge because an attacker can impersonate a user by simulating her communications latency. As detailed in Section 2.4.4, this challenge has been addressed by inserting a special device, delay mask, in the round-trip path between the client and its authenticator. The goal of the DM is to prevent any entity, but the authenticator, from being able to estimate $NRTT$. This marks the two differences between $NRTT$ and the state-of-the-art authentication factors. The first difference is that the client does not know the value of its $NRTT$, which makes it resilient to client compromise. The second difference is that the value of $NRTT$ is not communicated to the server, but rather is computed by the server. Deatiled

Research Challenge II Network instabilities may cause communications latency to vary and consequently result in legitimate login failures, which leads to poor user experience. Specificall, network communications latency may vary due to different reasons including congestion, queuing delays, server load, contention ratio in local network, and ISP throttling or traffic shaping operations. Therefore, the naive measurement of $NRTT$ may result in poor performance if such network instabilities are not carefully handled. Our $NRTT$-based authentication factor is designed with such instabilities in mind and hence it incorporates the necessary measures to alleviate the impact of such instabilities. In the following, we classify network instabilities into three categories, namely, instantaneous instabilities, long-term instabilities, and routing instabilities, and in Section 2.4, we show how to mitigate the impact of each category on the measurement of $NRTT$.

Instantaneous instabilities are instabilities which lead to transient changes in communications latency and hence, it only affects a few of the profiling signals. This type of instability is the most common one and can be addressed through outlier filtering (**Algorithm 2**) as detailed in Section 2.4.3.

Long-term instabilities are instabilities that stay long enough to affect all or most of the of profiling signals, however, they are not permanent. Such instabilities are mainly caused by low bandwidth, congestion, or variable traffic volume at the location of the user (i.e., the local network segment connecting the user to the network backbone). For example, if a user has a low bandwidth Internet, she will experience longer communications latency while her roommate is watching an HD movie on-line. We address this category of network instabilities by establishing multiple profiles, as detailed in Section 2.4.5.

Routing instabilities are instabilities that result in permanent changes in network communications latency due to, for example, permanent network routing changes. We introduced a dynamic temporal profiling technique (re-profiling the user every week based on the most recent login instances) to address long term routing instabilities. The main motivation of this profiling technique is to capture and handle changes in $NRTT$ over extended time periods. Changes of the $NRTT$ over long time periods are mainly due to routing changes in the Internet. We have designed the temporal profiling algorithm to handle such fairly uncommon cases. The design of our temporal profiling algorithm (such as the selection to profile every week) is guided by the long line studies of the routing behavior over the Internet. For example, earlier studies ([81] [109] [82] [114]) showed that most of the important IP prefixes have stable routes and that instabilities only exist in a small portion of the global Internet. A recent study [38], which was based on 3-year daily data and 8-year monthly data, confirms the results of the earlier studies and further shows that routing changes have strong weekly periodicity and the rate of change in routing decisions is stable over time, despite of the overall growth in the size of the Internet. Furthermore, it also reveals that only a small fraction of ASes are responsible for the vast majority of routing changes. Additionally, the 2013 experimental data of [80] indicates that only 2 out of 150 PlanetLab nodes showed 1-hop change while the others remain unchanged and the latency variance due to the 1-hop change is negligible. Therefore, routing instabilities are not common and can be addressed, like other unexpected events (e.g., forced traffic-reroute, login during DoS attack), using the backup failure techniques discussed in Section 2.4.6.

2.4 A Use Case of Two-factor Authentication System with $NRTT$: Design and Implementation

In this section, we demonstrate the design and the implementation of secure and reliable $NRTT$-based authentication factor through a use case of two-factor authentication scheme, dubbed AMAN. AMAN uses traditional passwords as the first authentication factor and $NRTT$ as the second authentication factor.

2.4.1 Assumptions

In the context of this work, we assume a powerful attack model, in which attackers already compromised traditional passwords of users. However, it is intuitive to conclude that AMAN, solely by itself, does not defend against perpetrators who both compromise legitimate user credentials and have access to her profiled location. Such attacks can be thwarted by augmenting AMAN with additional authentication factors, such as browser fingerprinting, as explained in Section 2.4.6. Additionally, Denial of Service (DoS) and remote access attacks (e.g., Rootkits and RATs on end user devices) are out of the scope of this work. We assume that the registration phase, when reference profiles are built, is secure, which is a reasonable assumption as it is required by all profiling schemes. We also assume that delay masks are secure and connected to network segments that are not accessible by attackers.

2.4.2 The Authentication Protocol

As depicted in Figure 2.2, the ecosystem of AMAN comprises three entities, users, web server (aka, authenticator or verifier), and delay masks. To demonstrate the authentication protocol in AMAN, we use the typical web-page login scenario depicted in Figure 2.3. The authentication protocol has two-phases, registration phase and login phase. The registration phase is a one-time process that initializes reference profiles, while the login phase is initiated with every login attempt to build real-time profiles. A profile is represented by the mean and the standard deviation of a number (N) of $NRTTs$ measured between user and server. Based on the distance between real-time profiles and reference profiles, the decision algorithm at the authenticator grants or denies access. To enhance the accuracy of reference profiles, they are updated with every successful login, that is, the new reference profile after a successful login is the combination of the current reference profile and the real-time profile of the last login. The new reference profile replaces the current one in server database while the real-time profile vanishes after the access decision is made. Using Figure 2.2, **Algorithm 1** presents the detailed steps of the registration phase and login phase, where each step in the algorithm maps to the corresponding step in Figure 2.3.

2.4.3 Computing Profile Parameters

As shown in steps #4 and #5 of **Algorithm 1**, the server measures $NRTT$ statistics of its users by sending out a set of small packets (similar to ping messages), dubbed as profiling signals. In current prototype implementation, the profiling signals are a set of UDP/TCP packets with user login session information in the payload (e.g., session ID, index number, user's IP address, etc.) An estimate of the number of profiling signals (N) that is required to establish a sufficiently accurate profile is discussed in Section 2.2.2. Before sending profiling signal S_i ($i \in [1, N]$), the server records its send time ($SndTime_i$). When the corresponding acknowledgment ($ack_i,\ i \in [1, N]$) is received, the server records the reception time ($RcvTime_i$). Then, the server computes the round trip time of each signal as $NRTT_i = RcvTime_i - SndTime_i$. After computing the ($N$) $NRTT$ values, the server calls **Algorithm 2**, which uses the scheme in [84] to first remove the outliers and then, it computes the mean and the standard deviation of the remaining $NRTT$ values. Filtering outliers aims at alleviating potential instantaneous network instabilities (Section 2.3).

2.4.4 Delay Mask

The naive measurement of $NRTT$ allows attackers to easily figure it out. For example, the attacker can simply ping the server from the location of the user. Ping packets provide excellent estimation of the $NRTT$ between the user and the server, and hence can be used to compute profile parameters. If the attacker learns the profile parameters of a user and if she is in a location with $NRTT$ less than that of the user, she can easily mimic the profile of the user; she simply adds appropriate delay before acknowledging the profiling signals. To address this important security concern, AMAN introduces a special one-way forwarding device, delay mask, in the route between the server and its clients. The DM is deployed and controlled by the server and is set to only relay profiling signals from the server to its users.

The main objective of the DM is to prevent any entity, except the authenticator, from being able to estimate $NRTT$. The DM achieves this by creating new path-segments in the round trip path between the server and its users. Therefore, the communications time over the newly created path-segments cannot be measured by outsiders. In Figure 2.2, consider DM1 for instance, the $NRTT$ can be computed as:

$$NRTT = D_{s1} + D_{1u} + 2 \cdot D_u + D_{us}$$

where $(D_u + D_{us})$ is the delay over the path-segments from the user to the server (R_u and R_{us}), and D_{1u} and D_{s1} are the delays over the path-segments from DM1 to the user (R_{1u}) and from the server to DM1 (R_{s1}), respectively. the attacker may be able to estimate $(D_u + D_{us})$ by pinging the server from the location of the user, however, it is not possible for her to figure out D_{1u} and D_{s1} due to the stealthy nature of DM and its one-way communication architecture. In other words, round trip cycles

Algorithm 1 AMAN Authentication Process

```
    Input: Total number of DMs ($D$), number of profiles per user ($r \leq D$), number of selected profiles per user ($m \leq r$), number of profiling signals per profile ($N$)
    Output: Grant or deny access
1:  Registration Request: User sends request to server
2:  Profiling Request: Server seeks the permission of the user
3:  Ready to Profile:
      • User accepts profiling request
      • Server randomly picks $r$ out of the $D$ available DMs. Each of the selected ($r$) DMs will be used to generate a different profile for user
4:  Sending profiling signals: Server,
      • Sends $r \cdot N$ profiling signals interleaved among the $r$ DMs. Let $d_s = \{d_1, d_2, ..., d_r\}$ be the set of selected DMs and let $p_s = \{S_1, S_2, ..., S_{r \cdot N}\}$ be the set of profiling signals. Signal $S_i$ in $p_s$ is forwarded through DM $j$ in $d_s$, where $j = i \mod r + 1$.
      • Records $SndTime_i$ of each $S_i$ ($i \in [1, r \cdot N]$)
5:  Acknowledgments: For $i \in [1, r \cdot N]$,
      • User sends $ack_i$ of $S_i$ directly to server
      • Server records $RcvTime_i$ of $ack_i$
      • Server computes $NRTT_i = RcvTime_i - SndTime_i$
    Compute profile parameters: For $j \in [1, r]$, if all the ($N$) $NRTT$ values of $d_j$ are ready,
      • Server calls Algorithm 2 to compute mean and standard deviation for each of the $r$ profiles
```

```
 6: Registration done: Server,
      • Chooses the $m$ most stable profiles out of the computed $r$ ones (i.e. the
        $m$ profiles that has the smallest standard deviation values) as the set
        of reference profiles for the user. This step helps in alleviating routing
        instabilities.
      • Server acknowledges user and returns
 7: Login Request: User sends login request to server
 8: Profiling Signals: Server repeats step #4 using $m$ instead of $r$
 9: Acknowledgments: Repeat step #5 using $m$ instead of $r$
10: Response to User: Server sends the result of calling Algorithm 4
```

can neither be established on the path-segment between the user and the DM nor on the path-segment between the DM and the server. Additionally, hiding the location of the DM (by hiding its IP) prevents estimates of the delays to the DM through measurement of the delay to close by entities. Therefore, the server is the only entity of AMAN that can measure profile parameters. We note here that legitimate users do not learn anything about their profiles or the profiles of other users. This not only makes $NRTT$ transparent to the users as they do not need to memorize or remember anything, but also prevents compromised users from breaching the security of AMAN.

Finally, we note that even in the worst case scenario in which $NRTT$ is disclosed, the attacker can impersonate the user (assuming her password is already compromised) only from locations that have similar or lower $NRTT$ values compared to that of the legitimate location, which reduces attack service and makes it harder. Additionally, as we see in the next section, the deployment of multiple DMs adds to the complexity and sophistication of such attacks and makes it highly unlikely because the attack can only succeed from locations that have lower $NRTT$ values for all the used DMs; which can be made difficult to achieve by careful deployments of the DMs.

2.4.5 Multiple Profiles

Delay mask represents a novel idea in the design of AMAN because it makes $NRTT$ measurement robust and extremely hard to manipulate. However, AMAN with single DM suffers from two main limitations in **the case of password compromise**. The first is its vulnerability to un-throttled guessing due to the low entropy (E) in $NRTT$ ($E \approx 10$ bits [71], the detailed mathematical analysis of $NRTT$ entropy is omitted for the sake of space). The second is the impact of network instabilities on the usability of AMAN due to the potential increase in the number of legitimate login failures. To address these limitations, AMAN deploys multiple DMs in different network locations. Multiple DMs are used to create multiple different profiles per user. **Algorithm 1** shows how AMAN generates multiple profiles using multiple DMs.

In the following, we demonstrate how multiple profiles can deter un-throttled guessing and alleviate network instabilities, then we present the authentication decision algorithm.

Defense Against Un-throttled Guessing: In general, un-throttled guessing is a brute force attack in which the attacker is allowed to try all possible credential combinations until she hits the right one. However, it is extremely hard for the attacker to try all possible combinations in AMAN due to physical limitations. Attacker can only simulate latencies that are higher than her own by appropriately delaying acknowledgments of profiling signals, that is, there is no possible way for an attacker to impersonate a user whose $NRTT$ is lower than her own.

Even when passwords are compromised, multiple profiles help in defending against un-throttled guessing by both expanding authentication sample space (i.e., the collection of all possible credential combinations) and by making guessing extremely hard (if not impossible) to perform. It is fairly easy to show that multiple profiles considerably expand single-profile sample space. Assume that the entropy of the single-profile sample space is E, then the entropy of the m-profile sample space is simply $m \cdot E$, because the profiles are independent.

More importantly, multiple profiling further reduces the set of possible users that the attacker can possibly impersonate. For an attacker to impersonate a user, her $NRTT_i$ through DM_i ($i \in [1, m]$) has to be faster than the corresponding $NRTT_i$ (i.e., through the same DM) of the user. If *any* of the $NRTT_i$ values of the user is faster than the corresponding one of the attacker, the attack definitely fails. The larger the number of DMs (m), the harder the attack can be performed. In fact, by carefully positioning the DMs, the attack can be made extremely hard to succeed.

Algorithm 2 Compute Profile Parameters

```
    Input: Set $R = \{NRTT_i, i \in [1, 2, ..., N]\}$;
    Output: mean and standard deviation
 1: procedure FILTER OUTLIERS
 2:     Computes the median value of $R$: $M = Median(R)$
 3:     Computes median absolute deviation (MAD) of $R$:
            $MAD = b * Median(abs(NRTT_i - M))$;
          Where $b = 1.5$ for normal distribution
 4:     Remove $NRTT_i > M + \tau \times MAD$ from $R$;
          Where $\tau = 2$ (moderately conservative)
 5: end procedure
 6: Return mean and standard deviation of $R$
```

Alleviating Long-term Instabilities: Network latency comprises delay in local network and backbone delay. The work in [44] shows that traffic congestion is the main cause of local network delay, and it only marginally affects backbone delay. It also shows that the main contributing factor of the backbone delay is the speed of light where the delay jitter is extremely low [44]. These results lead to the conclusion that backbone network is much more stable than local network. The empirical results in [80] and our experiments also support this conclusion, that is, the main contributor of long-term network instabilities is traffic congestion in local networks. We design here a novel algorithm based on multiple profiling to mitigate the impact of such instabilities on $NRTT$ measurements.

To see that, consider the DMs depicted in Figure 2.2, which are used to establish different user profiles. All the profiles share the same local network segment (R_u) but have different and more stable backbone routes ($R_{s1} + R_{1u}, \cdots, R_{sm} + R_{mu}$) [44]. Therefore, congestion in the local network (R_u) will introduce similar noise in all the real-time profiles. To filter out such noise, AMAN uses **Algorithm 3**. AMAN first measures the difference between real-time mean (x) and reference mean (μ) of each profile as $\triangle T_i = x_i - \mu_i$, where $i \in [1, m]$. Then, AMAN verifies that (i) either all the $\triangle T_i$ values are greater than δ or all the $\triangle T_i$ values are less than $-\delta$, where δ is the error tolerance defined by Equation (1) in Section 2.2.2, and (ii) the variance of the $\triangle T_i$ values is less than $\eta = 0.5$, where η is an experimentally predefined value. If these two conditions are true, then it is highly likely that the noise is caused by local congestion. In this case, AMAN simply subtracts the average noise value ($\triangle T = mean\{\triangle T_i\}$) from the mean of each real-time profile before applying the authentication decision algorithm (Section 2.4.5).

The Access Decision Algorithm: The access decision algorithm (**Algorithm 4**) presents the logic by which AMAN grants or denies access to end users based on the real-time profiles and the stored reference profiles. As explained in Section 2.4.2, the server keeps the m most stable profiles as the reference profiles for each user. With each login attempt, the server builds m corresponding real-time profiles. After

Algorithm 3 Filter out Long-term Instability

```
    Input: real time means $x_j$; reference means $\mu_j$, $\delta$, $\eta$
    Output: shared increment $\triangle T$
 1: procedure CALCULATE-INCREMENTS
 2:     Initialization: $\triangle T_0 = 0$
 3:     for $j \in [1, 2, ..., m]$ do
 4:         $\triangle T_j = x_j - \mu_j$
 5:         if $\triangle T_j \cdot \triangle T_{j-1} < 0$ (opposite trend) then
 6:             return $\triangle T = 0$ ;
 7:         end if
 8:         if $-p < \triangle T_j < p$
    (within the error tolerance of the Gaussian distribution) then
 9:             return $\triangle T = 0$ ;
10:         end if
11:     end for
12:     $std$ = standard deviation$\{\triangle T_j\}$
13:     if $std > \eta$
            (increments vary a lot) then
14:         return $\triangle T = 0$;
15:     end if
16:     $\triangle T$ = mean$\{\triangle T_j\}$
17:     return $\triangle T$
18: end procedure
```

computing the mean (μ_i) and the standard deviation (σ_i) of each of the m real time profiles (Section 2.4.3), AMAN uses the Gaussian PDF algorithm [123] to compute the distance ($score_{PDF}$) between the reference profiles and the the real-time profiles:

$$score_{PDF} = \frac{1}{m} \sum_{i=1}^{i=m} e^{-\frac{(x_i-\mu)^2}{2 \cdot \sigma^2}}$$

Access is granted if $score_{PDF} \geq Threshold_{PDF}$, otherwise access is denied. The threshold is experimentally selected to match the desirable trade-off between false positives and false negatives as detailed in Section 2.5.

To further refine and tighten the access decision, AMAN can use out-of-band channels such as SMS or email to request supporting evidence in doubtful or borderline situations. In this case, the decision will be either clear accept, clear deny, or supporting evidence is required. The supporting evidence could be a random number delivered to the user through an out-of-band channel. This enhancement serves multiple purposes: (i) it can alleviate legitimate login failures that may occur due to unexpected events on the Internet such as network traffic re-route, (ii) it can support arbitrarily mobile clients, (iii) it can be used as a backup channel to recover from long-term login failures such as exceeding the maximum number of login retries, and (iv) it can be used to establish new spacial profiles for the user in new login locations.

2.4.6 Other Important Discussions

The cyber security threat landscape is complex and continuously evolving, and hence, no security mechanism is foolproof. In this section, we discuss the limitations of the proposed scheme and provide corresponding solutions, including: (1) the potential integration of AMAN with other authentication factors to support arbitrarily mobile clients; (2) defense against sophisticated attacks, in which attackers both compromise legitimate credentials and have access to the legitimate login locations; and (3) defense

Algorithm 4 Authentication Decision Algorithm

```
    *Input:* real time parameters $(x_j, y_j)$
             reference parameters $(\mu_j, \sigma_j)$
    *Output:* authentication decision: True or False
 1: procedure COMPARE-PROFILES
 2:     $\triangle T$ = call Algorithm 3
 3:     $score_{PDF} = 0$
 4:     for $j \in [1, 2, ..., m]$ do
 5:         $x_j = x_j - \triangle T$
 6:         $score_{PDF} += e^{-\frac{(x_j - \mu)^2}{2 \cdot \sigma_j^2}}$
 7:     end for
 8:     $score_{PDF} = score_{PDF}/m$
 9:     if $score_{PDF} \geq threshold$ then
10:         return True $\leftarrow$ Grant access
11:     else
12:         return False $\leftarrow$ Deny access
13:     end if
14: end procedure
```

against low rate DDoS attack; and (4) considerations of the deployment for the delay masks.

Mobility and Login Failures: Mobile clients can be broadly classified into: (i) ***Low mobile clients*** who frequently login from a number of locations such as home, office, and library. In this case, AMAN simply creates a separate profile for each location. A user will be granted access if her real-time login profile matches any of the stored profiles. (ii) ***Arbitrarily mobile clients*** who may login from any arbitrary location. In applications that support such mobile clients (also in case of login failure), AMAN could use other authentication factors such as browser fingerprinting or random strings delivered through out-of-band-channels such as SMS and emails and then apply the enhanced decision algorithm as detailed in Section 2.4.5. However, we note here that if other factors are used to enable arbitrary mobility, mobile users do not benefit from the added security features provided by $NRTT$ such as resiliency to Phishing and MitM.

Sophisticated Attackers: As clarification, a sophisticated attacker, in our attack model, is that an attacker who compromised the password, knows the login location of the user, and has physical access to the login location of the user. We acknowledge that such attacker may succeed in impersonating the user, however, we would like to highlight the following facts: (1) We do not and can not claim that our authentication mechanism can be used anonymously for any generic application, but rather we have clearly stated that our $NRTT$ based authentication mechanism could greatly enhance the security of authentication in certain applications such as location-based authentication applications. Similar to most of the security mechanisms, we acknowledge that persistent and targeted attacks (attackers attempt to detect and exploit individual details of users) are very challenging and hard to block [83]. However, we note that, most of the authentication-based cyber attacks (e.g., stealing username/password by compromising the database of the web server) do not incorporate the users' location information and simply try to use these credentials as soon as possible from arbitrary locations, before being revoked. Therefore, we are confident that even though our authentication mechanisms is not completely fool-proof, it decreases the probability of attack success by greatly reducing the attack surface from anywhere in the world to only the login location of the user. (2) The "login location" as defined in our work does not mean the geographical nearby places but the same login network. For example, $NRTT$ will vary substantially if the user logs in through different networks (e.g., 4G, WiFi, etc.), even in the same room. (3) As mentioned in this work, the login location issue can be addressed by augmenting our authentication mechanism with additional factors such as browser fingerprints and on-demand dynamic passwords.

Low Rate DDoS Attacks: DDoS attacks are out of the scope of this work. However, instead of completely taking down the service, low rate DDoS attacks increase the network traffic (i.e., increase the $NRTT$) and hence, it may hinder the intended functions of our authentication mechanism. This type of attacks may occur in different scenarios and hence can be alleviated according to each specific scenario: (1) Low rate DDoS attack in the server or the local network of the user is implicitly addressed and is already alleviated by the proposed shared increment removal algorithm utilizing the deployment of multiple DMs. This is simply because Low rate DDoS attacks in this scenario cause similar delays for all the real time profiles and hence can be easily filtered out; (2) Low rate DDoS attack against one of the DMs is alleviated by the design of multiple profiles and the use of the adaptive decision algorithm. For example, for a design with 3 DMs, even if one of the DM paths is under low rate DDoS attack, the overall output of the decision algorithm wouldn't be affected much because it is an averaging of all the three profiles. In addition, the adaptive decision algorithm will warn the server if one of the profiles goes beyond a certain threshold while the others remain at normal level; (3) The probability of low rate DDoS attack in multiple DM paths with different rates is very low, however, it still can be addressed by randomising the DMs per session per user.

Considerations for the Deployment of the DMs: Recall that the DM is assumed to be connected to a separate dedicated secure network segment such that attackers (and also legitimate users) do not know its IP and can neither connect from the location of the DM nor can they compromise it. One way of hiding the IP of the DM is by spoofing the IP address of the server. Before forwarding the profiling signal, the DM sets the source IP field of the packet to the IP address of the server instead of its own IP. In addition to hiding the existence of the DM, this also makes AMAN transparent to end users. However, controlled spoofing of the IP address of the server by the DM is very challenging because it requires collaboration with the ISP hosting the DM to avoid dropping of the spoofed packets.

In addition, Software Defined Network (SDN) provides the possibility of a lightweight solution due to its centralized control plane design. The ISP controller could easily modify the forwarding policy globally and route the packets arbitrarily.

Furthermore, the deployment cost of multiple DMs could be reduced by utilizing existing infrastructure of the web service. For example, Google has many offices, repair branches, and data centers where DMs could be deployed.

2.5 Evaluation of $NRTT$ As an Authentication Factor Using AMAN

We conduct a thorough set of experiments to evaluate the usability (in terms of false negative rate, FN) and the security (in terms of false positive rate, FP) trade-offs of AMAN. We also study the impact of network instabilities on AMAN and assess its performance overhead. Specifically, we measure the following five metrics: (i) the false negative rate (FN), which is the probability that a legitimate login attempt fails, (ii) the false positive rate (FP), which is the probability that a perpetrator who *possesses the password* of a legitimate user successfully authenticates on her behalf, (iii) the login latency overhead (LLO), which is the average extra time it takes a user to successfully authenticate in AMAN compared to legacy one-factor password authentication, (iv) the storage overhead (SO), which is the extra storage space required per user in AMAN, and (v) the bandwidth overhead (BO), which is the extra network bandwidth incurred per login instance in AMAN. For all these metrics the lower the value is the better.

We measure the variations in these metrics by varying four parameters: (i) the number of profiles per user (m), (ii) the decision threshold (DT), (iii) the number of profiling signals (N), and (iv) the maximum number of login retries (LR). In all the following experiments, unless otherwise stated, we use $N = 45$, $LR = 2$, $m = 2$, $DT = 0.85 : 0.005 : 0.92$, and Gaussian PDF in the access decision algorithm. We re-emphasize that the FP values presented here are for powerful attackers, that is, attackers who already know passwords of legitimate users they try to impersonate.

2.5.1 Experimental Setup

We build a test-bed that implements the three entities of AMAN with the following configurations:

- **Users**: The user population consists of 10 Amazon EC2 instances, 130 PlanetLab nodes, 25 GENI nodes, and 63 residential WiFi users randomly selected from different places in USA and Canada.
- **Authenticators**: The authenticator runs Apache HTTP Server version 2.4 and a web service implemented in HTML and PHP. The authentication credentials (username, password, profile mean, profile standard deviation, decision threshold, etc.) are stored in a *MySQL* database.
- **Delay Mask**: We configure 3 PCs to work as the DMs. A *C* program is implemented on top of the MAC layer to relay the profiling signals from authenticators to users. The DMs are deployed in Oregon (USA), Germany, and Qatar.

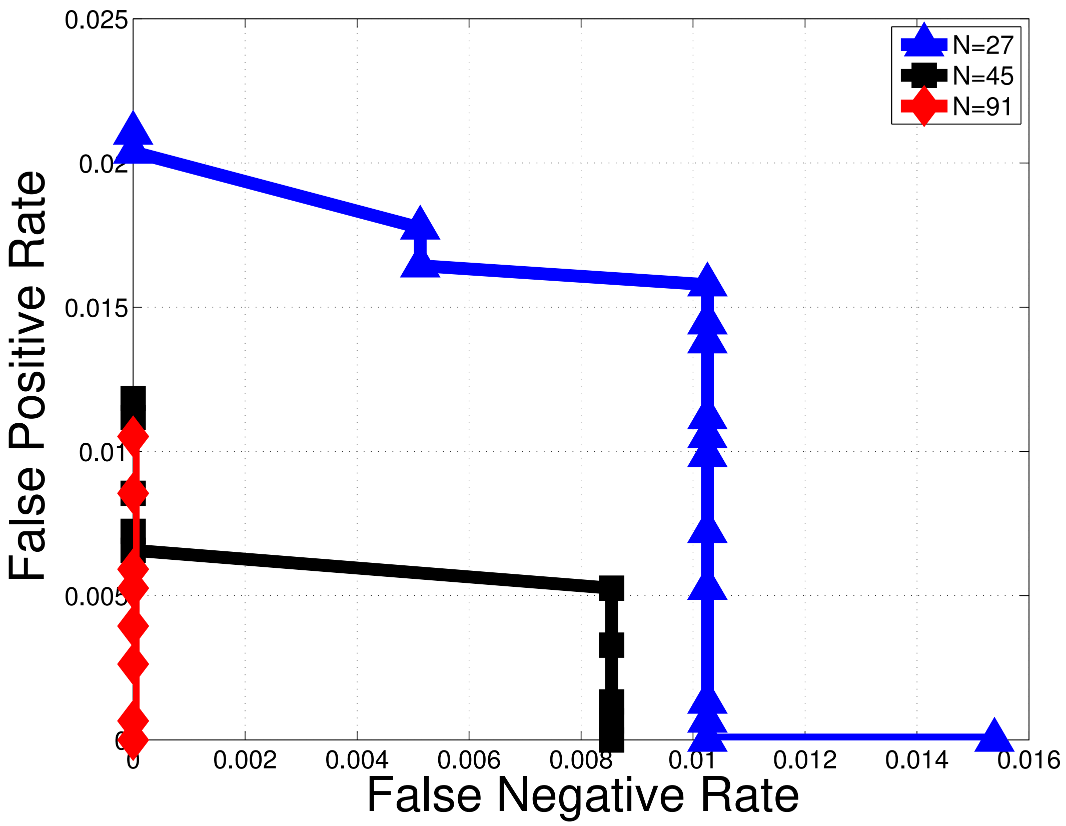


Figure 2.7 ROC curves of FP-FN; Varying the # of profiling signals N; $P = 2$, $LR = 2$, $DT = 0.85 : 0.005 : 0.95$.

2.5.2 Experimental Results

We first present the usability and security trade-offs, then we present the impact of traffic conditions on the access decision algorithm, and finally, we analyze the storage and performance overhead of AMAN.

Usability and Security Trade-offs: In this section, we study the trade-offs between usability (FN) and security (FP) properties of AMAN under different system parameters.

To measure FN, for each test instance (e.g., a test with $P = 2$, $N = 45$, $DT = 0.92$, $LR = 2$), each of the 25 GENI nodes and 63 residential WiFi users

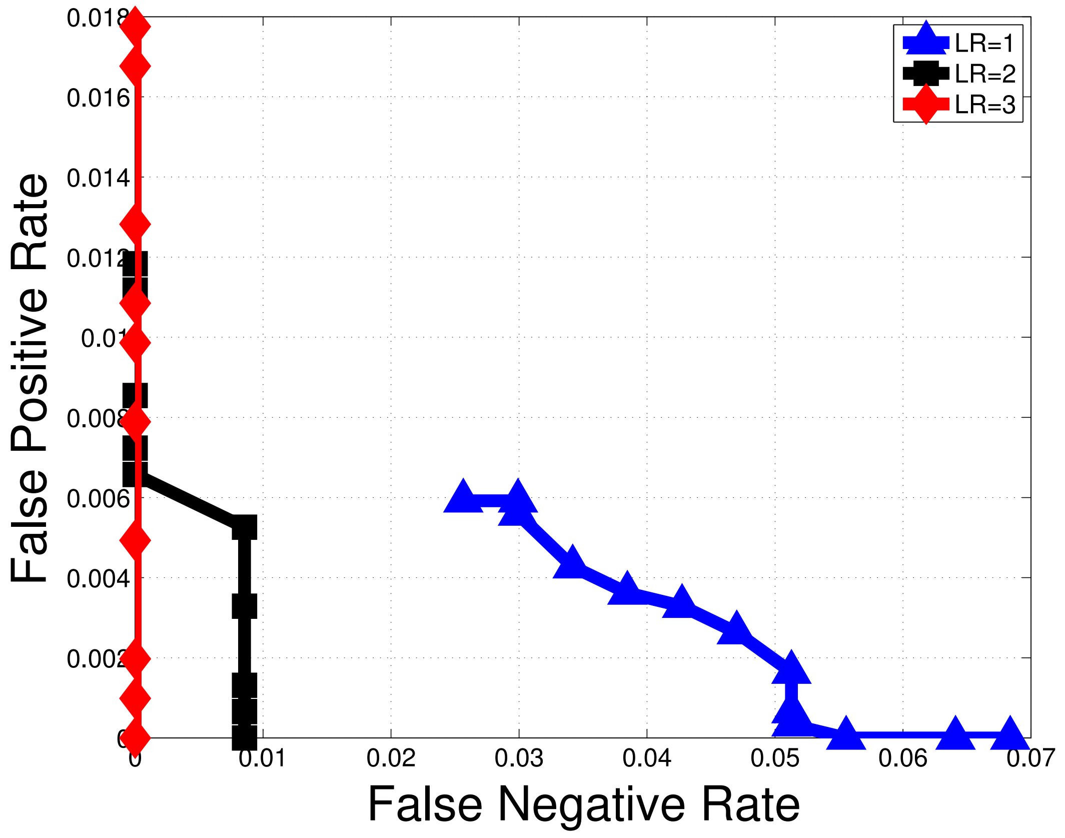


Figure 2.8 ROC curves of FP-FN; Varying the maximum # of login retries LR; $N = 45$, $P = 2$, $DT = 0.85 : 0.005 : 0.95$.

attempted to login 300 times. The data were collected within one day period for GENI nodes (i.e., wireline connections). For residential users (i.e., WiFi connections), the data were collected at random time within one month period). To measure FP, each of the 25 GENI nodes tries to login 300 times within one day period using the username and password of the other 217 clients (i.e., the other 24 GENI nodes, 63 residential users, and 130 PlanetLab nodes), a total of 1,627,500 (25*217*300) impersonation attempts are launched for each test instance. In all the following experiments, we vary the DT from 0.85 to 0.92 with 0.005 step and measure FP and FN values for each threshold point.

Note that, the experiments are indeed conducted over long time periods, not within a few days. The experimental data for the 63 residential users was collected and

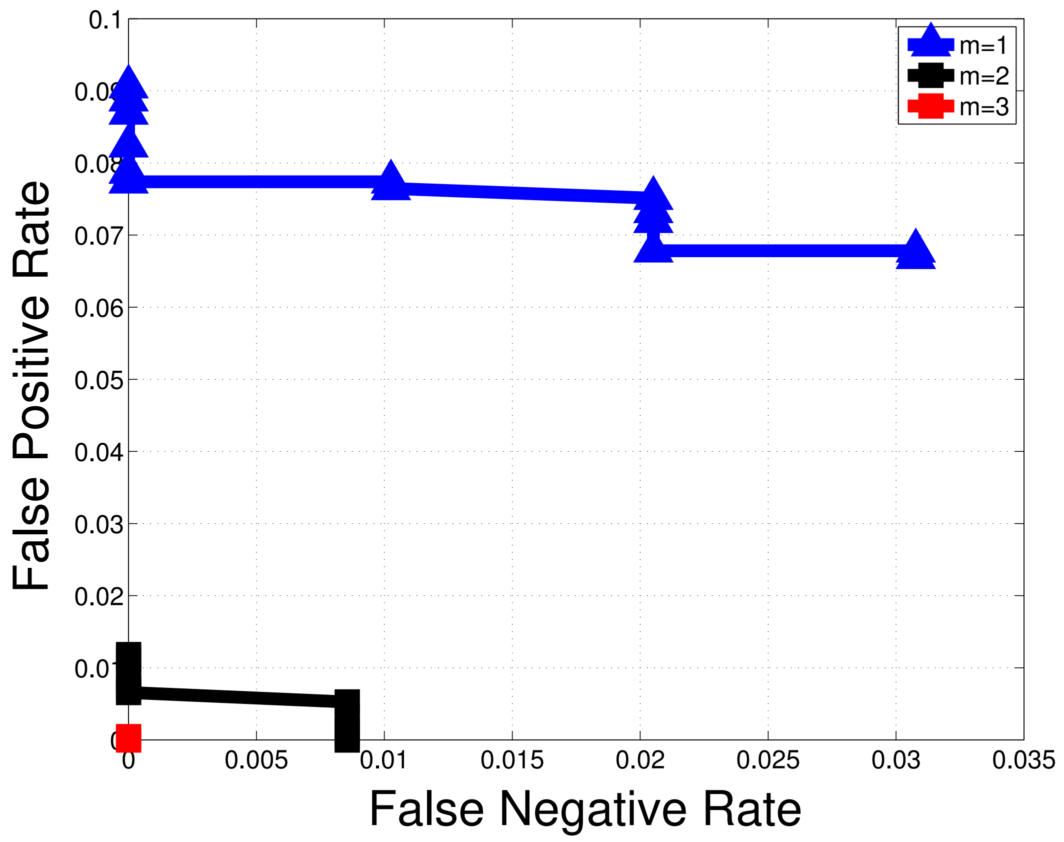


Figure 2.9 ROC curves of FP-FN; Varying the # of profiles per user P; $N = 45$, $LR = 2$, $DT = 0.85 : 0.005 : 0.95$.

tested over one month period. In addition, our testbed has been running for about one year. Even though the tests are not performed continuously, they still demonstrate good performance for each individual test, thanks to the proposed dynamic temporal profiling algorithm described in Section 2.3.

Varying the number of profiling signals (N): Figure 2.7 shows the ROC (receiver operator characteristics) curves of FN and FP with variable DT for N equals 27, 45 and 91. The figure emphasizes the analysis results in Section 2.2.2 as it clearly shows that both FP and FN improve (lower values) as N increases. For example, AMAN can achieve $FP = 0$ and $FN \approx 0.85\%$ for $N = 45$. The figure also shows that with $N = 91$, AMAN achieves $FP = 0$ and $FN = 0$.

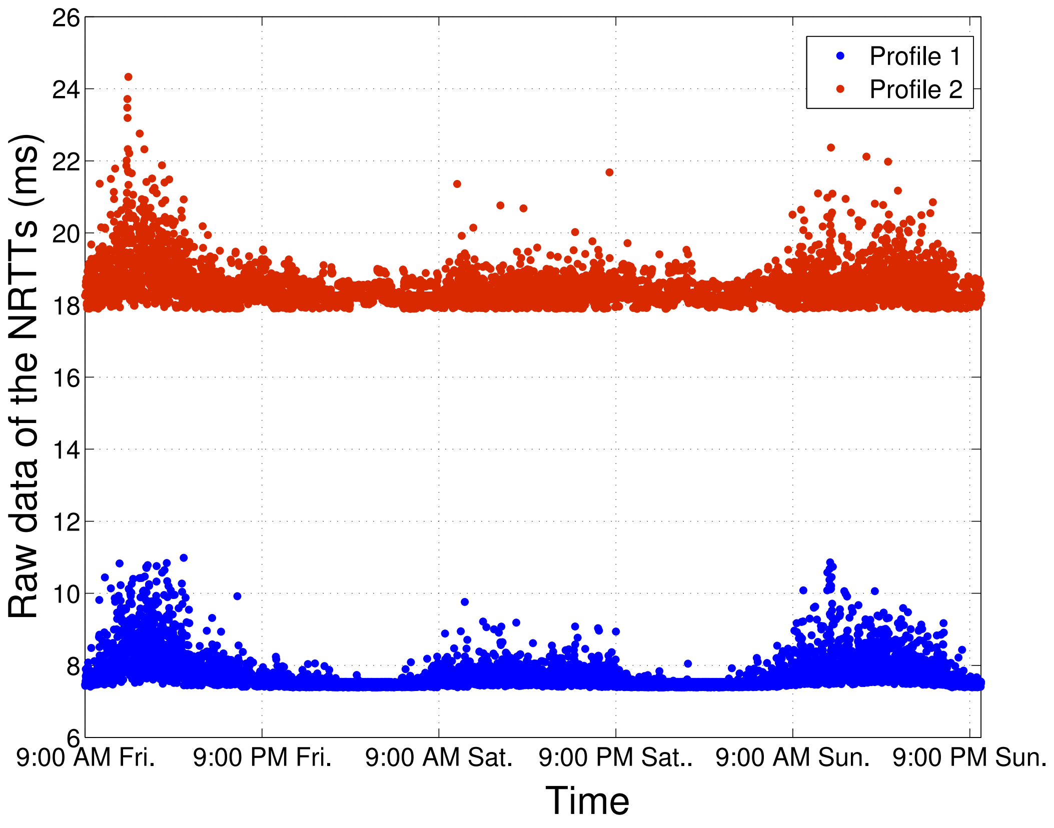


Figure 2.10 Raw data of $NRTTs$ for two different profiles taken every 20 minutes for 60 hours.

***Varying the maximum number of login retries** (LR)*: Figure 2.8 shows the ROC curves of FN and FP with variable DT for LR values of 1, 2, and 3. The figure clearly shows that LR has big positive impact on FN. When the user has a second chance to login after the first failed one, FN can be significantly reduced. On the other hand, LR has a relatively very small negative impact on FP. Increasing LR, only slightly increases FP because attacker almost gains nothing when given another login chance, even if the new login chance is performed from a new location, thanks to the large authentication sample space of AMAN. The figure shows that for $LR = 3$, AMAN can achieve both $FN = 0$ and $FP = 0$.

***Varying the number of profiles per user** (m)*: Figure 2.9 shows the ROC curve of the FN and FP with variable DT for m values of 1, 2 and 3. The figure

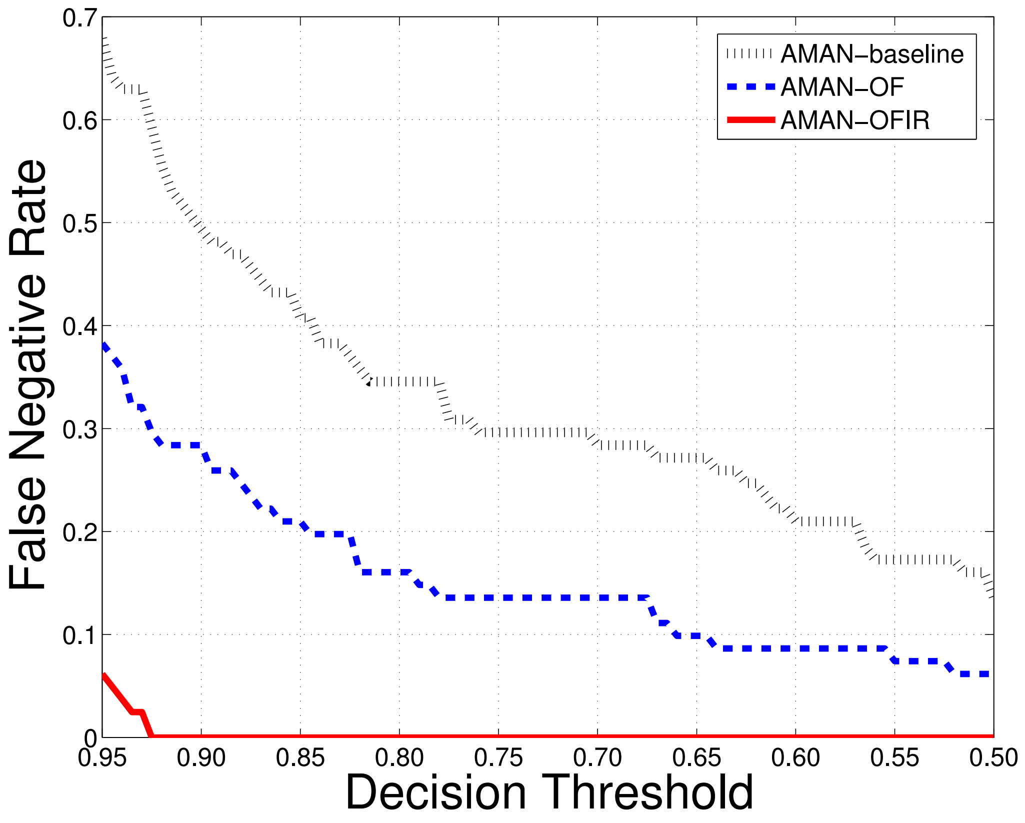


Figure 2.11 The FN curve with variable DT for three different decision algorithms.

clearly shows that the number of profiles per user has significant positive impact on FP. The 1-profile FP value is much larger than that of the 2-profile and the 3-profile cases. This is intuitive because multiple profiles both increase the authentication sample space, and considerably decrease impersonation by physically limiting the number of locations from which impersonation could be lunched (Section 2.4.5). Similarly, the larger the number of profiles, the lower the FN. Recall that multiple profiles help to filter out network instabilities and hence decreases FN. The figure shows that with $m = 3$, AMAN can achieve $FP = 0$ and $FN = 0$.

Impact of Network Instabilities: For this experiment, we set a WiFi user connected to the university campus to login every 20 minutes for 60 hours, from 9AM Friday through 9PM Sunday. Figure 2.10 shows all the $NRTT$ values measured

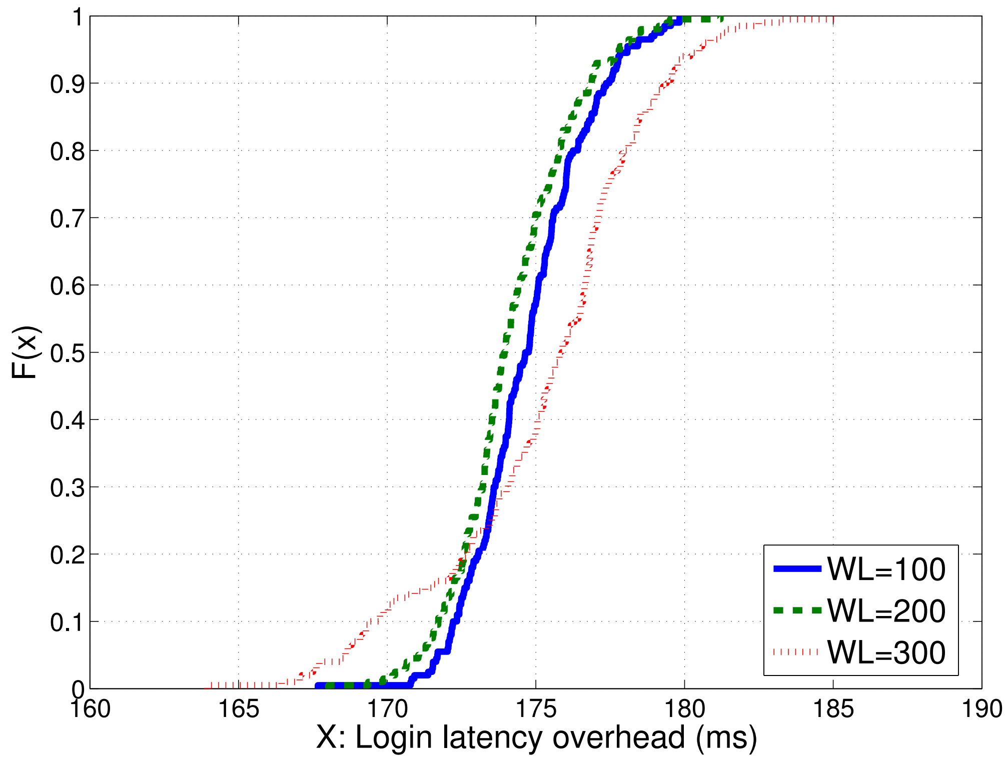


Figure 2.12 The CDF of login latency overhead with variable WL.

for two different profiles during the test period, a total of 8100 (60*3*45) $NRTTs$ per profile. The figure clearly shows the variations in $NRTT$ values over different time periods. To evaluate the impact of such variations on the FN rate and to assess the capabilities of AMAN to cope with them, we use the data in Figure 2.10 to compute the FN rate over all the login attempts. Recall that AMAN uses outlier filtering (**Algorithm 2**) and shared increment removal (**Algorithm 3**) to alleviate instantaneous and long-term instabilities, respectively. Figure 2.11 shows FN as a function of DT for baseline AMAN (no filtering), AMAN with outlier filtering alone (AMAN-OF), and AMAN with both outlier filtering and shared increment removal (AMAN-OFIR) with $p = 0.5ms$ and $q = 0.5ms$. The figure shows that network instabilities can badly hurt the usability of AMAN as evidenced by the 10% FN rate, which is the best FN rate that baseline AMAN can achieve. On the other hand, the figure shows the effectiveness of AMAN in coping with network instabilities as evidenced by the 0 FN rate achieved by AMAN-OFIR, almost irrespective of the DT value used. The flexibility in DT is very important as it allows AMAN-OFIR to achieve lower FP rate as well, because the higher the DT, the better the FP.

2.5.3 Performance Overhead

Login Latency Overhead (*LLO*): In this experiment, we evaluate the login latency overhead (LLO) per user in AMAN compared to that in legacy password authentication. We use the 10 Amazon instances in our test-bed to generate variable server workload (WL), which is measured by the number of login requests per second. Then, we measure the LLO of 12,420 logins under server workloads of 100, 200 and 300 login requests per second. Figure 2.12 shows the empirical cumulative distribution functions (CDFs) of LLO under each WL. The figure shows that more than 99% of the login instances have LLO less than 0.185 seconds, which is unnoticeable to humans.

Bandwidth Overhead: The bandwidth overhead (BO) is caused by the profiling signals and the acknowledgments of every login attempt. Each of the m profiles requires N profiling signals and N acknowledgments per login attempt. The size of the profiling signal/acknowledgment is about 50 bytes including all the headers. Therefore, $BO = 100 \cdot m \cdot N$ bytes/login instance. Based on our experiments, $m = 3$, $N = 45$ provides excellent trade-off between FP and FN, and hence, $BO = 100 \cdot 3 \cdot 45 = 13500 \approx 13K$ bytes. This is a negligible overhead given the fact that login bandwidth consumed by most of the popular websites (e.g., Chase.com, Facebook.com, Amazon.com) ranges between 10.98K and 125.47K bytes, according to a study that we have conducted on 12 popular web services.

2.6 Evaluation of $NRTT$ and Other State-of-the-art Second Authentication Factors

In this section, we perform thorough comparative evaluation of $NRTT$-based authentication factor against state-of-the-art authentication factors by employing a famous benchmarking framework for web authentication, which has been proposed by Joseph Bonneau et al. [29]. We compare $NRTT$ (represented by **H** in Table 2.2) against: **A1**: Second Authentication Code (SAC) through email [46], **A2**: SAC through mobile SMS/voice [53], **B1**: disconnected hardware token (e.g., RSA SecurID [14]), **B2**: connected hardware token (e.g., smartcard-like USB token, NFC, etc. [50] [39]), **C**: paper token (e.g., PIN+TAN [134]), **D**: biometrics (e.g., fingerprint, iris, voice recognition, etc. [112]), keystroke [95], **E1**: non-keystroke device characteristics (e.g., mouse click pattern, [105]), **E2**: hardware/system signature (e.g., browser fingerprint, trusted platform module, etc. [96] [70] [43]), **F**: knowledge-based information (e.g., photo recognition of somebody you know [30]).

The benchmark encompasses twenty five properties grouped into three categories, namely, **usability**, **deploy-ability** and **security**. Each property has two-dimensional score (VU). The first entry of the score (V) indicates whether the authentication scheme offers the property ($V = Y$), does not offer the property ($V = N$), or partially offer the property ($V = A$). The second entry of the score (U) indicates whether this property offering is better than ($U = " + "$), similar to ($U = " \quad "$, i.e., no symbol), or worse than ($U = " - "$) that of the legacy password scheme (the baseline). For example, if an authentication scheme offers a certain property better than the baseline, the score of this authentication scheme against that property will be $Y+$. Table 2.2 presents the comparison among factors. The results clearly show that $NRTT$ has the best combination of usability, deploy-ability and security proprieties. In the following, we explain the benchmark properties that may not be straightforward and drop the explanation of the intuitive ones.

Table 2.2 Evaluation of *NRTT* and Other State-of-the-art Second Authentication Factors

		A1	**A2**	**B1**	**B2**	**C**	**D**	**E1**	**E2**	**F**	**G**	**H**
Security	Resilient-to-Physical-Observation	A+	A+	A+	Y+	N-	Y+	N-	N-	Y+	Y+	Y+
	Resilient-to-Targeted-Impersonation	A+	A+	Y+	Y+	Y+	N-	Y	Y	A+	N-	A+
	Resilient-to-Throttled-Guessing	Y	Y	Y	Y	Y	Y+	Y	Y	Y	Y	Y
	Resilient-to-Unthrottled-Guessing	Y	Y	Y	Y	Y	Y+	Y	Y	Y	A-	Y
	Resilient-to-Internal-Observation	N	N	Y+	N-	N-	N-	N-	N-	N-	Y	Y+
	Resilient-to-Leaks-from-Other-Verifiers	Y+	Y+	Y+	Y+	Y+	N-	N-	N-	N-	Y	Y+
	Resilient-to-Phishing	N	N	N	N	N	N	N	N	N	N	Y+
	Resilient-to-Theft	Y	N-	N-	N-	N-	N-	Y	Y	Y	Y	Y
	No-Trusted-Third-Party	N-	N-	N-	N-	Y	Y	Y	Y	Y	Y	N-
	Requiring-Explicit-Consent	Y	Y	Y	A-	Y	Y	Y	Y	Y	Y	Y
	Unlinkable	N-	N-	Y	Y	Y	N-	N-	N-	N-	A+	Y+
Usability	Memorywise-Effortless	A+	A+	A+	Y+	Y+	Y+	Y+	Y+	Y+	N	Y+
	Scalable-for-Users	Y+	Y+	N-	N-	Y+	Y+	Y+	Y+	Y+	N-	Y+
	Nothing-to-Carry	Y	N-	N-	N-	N-	Y+	Y+	Y+	Y+	Y+	Y
	Physically-Effortless	N	N	N	A+	N	N-	N	Y+	Y+	N-	Y+
	Easy-to-Learn	Y	Y	A-	Y	A-	Y	Y+	Y+	Y+	Y	Y
	Efficient-to-Use	N-	N-	A-	A-	A-	N	Y	Y	Y+	N-	Y+
	Infrequent-Errors	Y	Y	Y	Y	Y	A-	A-	N-	Y	A-	A-
	Easy-Recovery-from-Loss	Y	Y	N-	N-	N-	N-	N-	N-	N-	Y	Y
Deployability	Accessible	Y	A	A	A	A	N-	Y	Y	Y	Y	Y+
	Negligible-Cost-per User	Y	N-	N-	N-	Y-	N-	Y	Y	A-	Y	Y
	Server-Compatible	N-	N-	N-	N-	N-	N-	N-	N-	N-	N-	N-
	Browser-Compatible	Y	Y	Y	N-	Y	N-	Y	N-	N-	Y	A-
	Mature	Y	Y	Y	Y	Y	N-	N-	N-	N-	A-	N-
	Non-Proprietary	Y	Y	N-	A-	Y	A-	A-	Y	N-	Y	Y

"-" = Worse than Legacy Password; no symbol = Same as Legacy Password; "+" = Better than Legacy Password;
Y = offer the property; N = does not offer the property; A = partially offer the property.

2.6.1 Security

Resilient-to-Physical-Observation: It evaluates the potential leakage of authentication credentials by physical observation of users during login. For $NRTT$-based authentication, even legitimate users can not learn or compute their own profile parameters (Section 2.4.4). Therefore, it is completely resilient to physical observation. On the other hand, SAC and legacy password schemes are susceptible to physical observation. Biometric factors based schemes are partially resilient to physical observation due to the potential capture of individual biometrics using special tactics, such as lifting fingerprints from the glass surface of scanners.

Resilient-to-Targeted-Impersonation: It evaluates the potential of targeted impersonation by capturing or simulating specific user authentication factors. $NRTT$-based authentication which has been shown to be robust and unforgeable factor (Section 2.4.4), and hence, $NRTT$ is completely resilient to targeted impersonation. Other factors, take biometric factor - fingerprint for example, it is obviously susceptible due to fixed fingerprint values.

Resilient-to-Internal-Observation: It evaluates whether the attacker can capture credentials by intercepting the input of the user inside her device. As detailed in Section 2.4.2, $NRTT$-based authentication does not require any user input. Therefore, it is highly unlikely, even for the legitimate users, to learn or compute their own $NRTT$. On the other hand, SAC is susceptible to internal observation since attackers might be able to intercept authentication codes in SMS/email.

Resilient-to-Leaks-from-Other-Verifiers: It evaluates whether specific user's credentials could be leaked across different verifiers. For $NRTT$, thanks to the DMs, profile parameters across different verifiers are independent, and hence, user profiles in one verifier are decoupled from her profiles in other verifiers. However, biometrics factors (e.g., fingerprint) are completely susceptible.

Resilient-to-Phishing: It evaluates whether man-in-the-middle (MitM) attackers can capture credentials. Active-Phishing describes the attack in which perpetrators use forged websites to capture authentication credentials (legacy password, SAC, fingerprints, etc.) and then, use them in real time to impersonate legitimate users. The concept of MitM does not apply in the case of $NRTT$ because its users do not send credentials that could be intercepted by attackers, instead credentials are measured at the server. Similar to legacy passwords, both SAC and biometric factors are obviously susceptible to MitM attackers.

Resilient-to-Theft: It evaluates the potential leakage of credentials through loss or theft of special authentication devices, such as authentication tokens. This is not applicable to $NRTT$ as it does not require any devices. On the other hand, SAC through mobile SMS and all token-based factors are susceptible to theft-related attacks.

Unlinkable: It evaluates whether colluding verifiers can determine user credentials on other verifiers. As explained in *leaks-from-other-verifiers*, $NRTT$ profiles across different verifiers are uncorrelated. However, biometric factors (e.g., fingerprints) are obviously linkable due to unique user biometrics.

2.6.2 Usability

Scalable-for-Users: It evaluates the burden on users who may have multiple accounts on different web services. For example, in legacy password systems, a user has to create a different password for each service. $NRTT$ can be scaled to any number of accounts per user without creating any extra burden. $NRTT$ profiles for the same user on different verifiers are independent and are measured and stored by the server.

Physically-Effortless: It evaluates whether the authentication factor requires physical (as opposed to cognitive) user effort beyond, say, pressing a button. $NRTT$-based authentication does not require any such effort. On the other hand, SAC users need to transcribe passwords from their phones/email into browsers and fingerprint users need to scan their fingerprints.

Efficient-to-Use: It evaluates the time it takes the user to successfully login. We show that extra login latency overhead of $NRTT$-based authentication scheme (compared to the baseline of legacy passwords) is less than $185ms$, which is negligible and is completely unnoticeable by end users. On the other hand, SAC users have to wait for SMS message to get authentication code, which may take a few seconds, and biometric factor scanning may take several seconds.

Easy-Recovery-from-Loss: It evaluates the easiness by which users regain the ability to login if the login credentials are lost or forgotten. Authentication based on $NRTT$ is completely transparent to users as they do not keep or memorize any authentication credentials. It is highly unlikely (as proved by the results of our extensive experiments) for a legitimate user to fail login after three trials. However, in the very rare case of sudden and permanent changes in profile parameters, $NRTT$-based authentication (AMAN as introduced in 2.4) generates new reference profiles after verifying the identity of the user using either: (i) out-of-band channels such as email or SMS, (ii) other complementary authentication schemes that may have been augmented with AMAN such as browser fingerprinting, (iii) if none of the previous options are available, a user can re-register with the web service as a new user. On the other hand, the loss of fingerprint credentials (e.g., physical damage to fingers) is permanent and impossible to be regained.

2.6.3 Deploy-ability

Accessible: It evaluates whether users may be hindered from using the scheme due to disabilities or any other physical conditions. Again, $NRTT$-based authentication scheme is completely transparent to users and is easily accessible irrespective of any disabilities. On the other hand, SAC and biometric factor based schemes are less accessible. For example, blind users cannot read SAC from phones, and users who broke their fingers cannot scan fingerprints.

Mature: It evaluates whether the scheme has been tested and deployed in large scale real-world scenarios. $NRTT$-based authentication (AMAN) is a new authentication mechanism and hence this property does not apply. However, AMAN has been extensively tested with relatively good size prototypes (155 non-mobile users and 63 residential WiFi users). SAC is widely deployed but biometric factor based authentication has not been used for remote authentication.

Non-Proprietary: It evaluates whether the scheme requires licensing due to intellectual property or royalties. Similar to SAC, $NRTT$-based authentication (AMAN) is an open framework that can be freely implemented by any interested entity. On the other hand, many biometric authentication schemes have patented hardware/software.

2.7 Related Work

2.7.1 Security Usage of $NRTT$

In [125], a packet delay-based scheme is proposed to detect man-in-the-middle (MitM) attacks. The delay is calculated using TCP packet headers with the assumption that the delay increases in the presence of MitM attacker. However, packet delay can easily be manipulated by, for example, pinging the network service.

2.7.2 IP Based Geolocating

Recently, Gmail has launched a service that enables its users to detect suspicious account login activities based on their IP information [8]. A suspicious attempt is detected by matching the relevant IP address(es) to a broad geographical location(s). However, IP-based verification (fixing a range of IPs) can be easily bypassed via (1) proxy-server; (2) VPN; (3) IP-hijacking. Many web clients are behind proxies (or VPN). The client and the proxy may be far apart. For example, the AOL network, which has a centralized cluster of proxies at one location (Virginia) for serving client hosts located all across the U.S. [101]. BGP/IP hijacking is much more common than current researchers think and it is hard to be detected in the form of local BGP hijacking [126] [48]. Furthermore, IP address based authentication suffers many other limitations due to current Internet infrastructure including: (1) extensive use of NAT, especially the use of Carrier Grade NAT (CGN) or Large Scale NAT (LSN) [57]; (2) complex and various IP address configuration policies by different ISPs. Therefore, IP address based authentication is more suitable for LAN other than general web service authentication.

2.7.3 Delay Based Geolocating

The authors of [67] propose a topology based approach to estimate the geographic location of a certain Internet host. Specifically, they ping the target from a set of landmarks. Then, based on the end-to-end delay, the target is considered closest to the landmark with the shortest latency. The primary result is that geolocation is more accurate when topology is considered (e.g., using traceroute, topology and per-hop latencies). Francis et al. [45] use traceroute command to collect round trip delay data then find the nearest host in terms of low latency. GeoPing[77] provides a list of end-to-end delays from a target to a set of geographic locations via ping command to reveal how close a target is to all the landmarks.

However, all of the methods mentioned above use naive measurement of the (e.g., Ping or traceroute) delay which is not practical on the Internet (e.g., probably to be blocked). In addition, none of these method consider network instabilities when using the latency data.

2.8 Conclusions

In this work, we proposed a novel, secure and usable web authentication factor based on Network Round Trip Time ($NRTT$) that strengthens the security of web service authentication by offering robust defenses against password compromise. We introduced a novel network component, delay mask, which turns $NRTT$ into a secure and robust authentication factor. More importantly, we designed and implemented various algorithms, with the help of multiple DM deployments, to alleviate network instabilities and expand authentication sample space of $NRTT$. The benchmark comparative results (Appendix 2.6) showed that $NRTT$ has superior security, usability, and deploy-ability properties among state-of-the-art authentication factors.

We designed, implemented and deployed a prototype for a use case of two-factor authentication (AMAN) with legacy passwords as first factor and $NRTT$ as a second factor. The experimental results showed that AMAN can achieve false positive and false negative rates as low as 0, while maintaining the login latency overhead below 185 ms.

In the future, we plan to make AMAN completely end user agnostic by replacing passwords with mechanisms that can automatically collect unique user characteristics.

Such techniques may utilize biometric tools such as typing behavior, screen resolution, device signatures and others.

CHAPTER 3

A NOVEL AND ROBUST PHISHING DETECTION FEATURE

3.1 Introduction

Phishing, one form of cyber-attacks, continues to be a growing concern not only to cyber security specialists but also to e-business users and owners. The severity of such cyber attack vector is continuously growing with the exponential increase in digital information generation and the increased reliance of people and business on cyber space. The Anti-Phishing Working Group (APWG) has seen rapid growth in the number of unique phishing websites detected from 2014 to 2016 [23]. The average annual growth rate is 97.36% and is expected to continue to grow. Estimates of annual direct financial loss to the US economy caused by phishing activities range from $61 million to $3 billion [58].

To mitigate the increasing damage caused by phishing, a broad range of anti-phishing mechanisms have been proposed over the past two decades. These anti-phishing techniques can be categorized into three broad groups [20]: (1) Detective solutions (e.g., website filtering); (2) Preventive solutions (e.g., strong authentication [69, 102, 43, 41, 42, 49, 68]); and (3) Corrective solutions (e.g., Site takedown [73, 72]). In this work, we focus on detective solutions. More specifically, we look at software-based phishing detection schemes that are specialized in identifying and classifying phishing websites. This class of approaches is arguably more important than other approaches because it helps in reducing human errors. Preventative and corrective solutions take a different approach, but if the user behind the keyboard has been successfully tricked by a phishing attempt, and willingly submitted sensitive information, then no firewall, encryption software, certificates, or authentication mechanism can help in preventing the attack from materializing [58]. Software-based phishing detection also delivers improved results compared to detection by user education (e.g., [78], [79], [116]) because phishing attacks normally aim at exploiting human weaknesses [75]. For example, a study of phishing detection using user education [115] shows a 29% false negative rate (FNR) for the best performance, while the software based approaches that are surveyed by the same study have FNR in the range of 0.1% to 10%. For this reason, we focus our study on software based phishing detection systems, and the term "phishing detection" will refer only to this form of detection in the rest of the dissertation.

In this work, we propose a new phishing detection feature based on the Network Round Trip Time, dubbed as $NRTT$. $NRTT$ has been introduced in [61] as a reliable and robust second web authentication factor. $NRTT$ simply captures the network round trip time that packets take in its journey from one Internet connected host to another and back to the original sender. We propose here a two-phase approach based on $NRTT$ to detect phishing web sites. In the first phase, the targeted victim site is identified through content analysis of the phishing website, URL characteristics, spam email content analysis, or combination of them. Victim website identification is well studied in literature (e.g., [87] [131] [90]), and hence we capitalize on the state-of-the-art mechanisms to build our victim identification component. In the second phase, we compare the average $NRTT$ values of both the phishing website and the targeted victim website from a certain vantage point. If the difference between the two $NRTT$ averages is greater than a threshold, the link is highly likely a phishing link. The intuition here is that the phishing website is likely to be hosted on a server different from that of the victim website, and hence, they will experience different average $NRTT$.

We summarize our contributions in this work as follows:

- Introduce a novel and highly usable Phishing detection feature based on Network Round Trip Time, $NRTT$.

- Propose a new phishing detection framework by carefully combining Phishing detection features based on the most recent Phishing trends and the analysis of state-of-the-art Phishing detection features.
- Present an algorithm to identify the target website from suspicious URLs.
- Provide comparative evaluation of the proposed scheme against state-of-the-art schemes and show that our scheme not only has comparable performance results but also can neutralize advanced Phishing attacks.

3.2 Background

3.2.1 State-of-the-art Phishing Attacks

In this section, we first present the various definitions of phishing, then we introduce some statistics about phishing between January 2010 and June 2016. Finally, we describe the phishing ecosystem.

What is Phishing? There is no consensus on how phishing should be defined. Different phishing definitions lead to different research directions and approaches (e.g., email filtering or website detection). It is important to clearly identify the target of any phishing detection approach to avoid confusion about its applicability in different scenarios. The target and scope of phishing detection approaches can be analyzed from the definition of phishing which has been adopted by such approaches. Therefore, presenting a background on the different definitions of phishing can help the readers understand the scope and the capabilities of different approaches. Table 3.1 summarizes the popular definitions of phishing. On one hand, the definitions of PhishTank [100], APWG [23], Xiang et al. [136], Tameshe et al. [106] cover the majority of cases in which phishers aim at stealing sensitive personal information such as authentication credentials. Table 3.2 shows the comparison of those phishing definitions based on phishing target and phishing strategy. The most dominant phishing strategies are social engineering (e.g., through fraudulent emails) and technical subterfuge (e.g., malware infection). However, sophisticated techniques

Table 3.1 Most Popular Definitions of Phishing

	Definition
PhishTank [100]	Phishing is a fraudulent attempt, usually made through email, to steal your personal information.
APWG [3]	Phishing is a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data & financial account credentials.
Xiang et al. [136]	Phishing is a form of identity theft, in which criminals build replicas of target Web sites and lure unsuspecting victims to disclose their sensitive information like passwords, personal identification numbers(PINs), etc..
Whittaker et al. [133]	A phishing page is any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party.
Khonji et al. [75]	Phishing is a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit.
Ramesh et al. [106]	Phishing is a fraudulent act to acquire sensitive information from unsuspecting users by masking as a trustworthy entity in an electronic commerce.

Table 3.2 Targets and Strategies of Phishing

	Target	**Strategy**
PhishTank [100]	Personal information	Social engineering
APWG [3]	Identity data, Financial account credentials	Social engineering
Xiang et al. [136]	Sensitive information	Not specified
Whittaker et al. [133]	Not specified	Not specified
Khonji et al. [75]	Not specified	Social engineering
Rameshe et al. [106]	Sensitive information	Not specified

(e.g., pharming [65]) are also used to harvest users' personal information from the Internet. On the other hand, the definitions of Whittaker et al. [133] and Khonji et al. [75] do not limit the attacker's target (e.g., sensitive personal information). They describe the phishing strategy (e.g., phishing website or socially engineered messages) without stating a specific phishing target (e.g., only state the attackers' benefit). To sum up, the definition of Whittaker et al.[133] is the most general among those reviewed, while APWG [23] defines the most commonly used phishing attacks in a specific manner.

How Does Phishing Work? In this section, we introduce the ecosystem of phishing in terms of phishing process, actors involved, their actions and interactions.

(i) Phishing Process: In a generic/traditional phishing scenario (i.e., massive email phishing campaigns), an attacker hosts a fake website, and presents users of a web service with convincing emails containing a link to the fake website. When a user of the web service opens the link and enters her sensitive data, data is collected by the server hosting the fake website.

As illustrated in Figure 3.1, Mihai and Giurea [94] suggest that a generic phishing process can be identified into five steps: (1) Reconnaissance: Phishers look for famous web service brands with a broad customer base; (2) Weaponization: Phishers design the selected phishing websites and social engineer on email spam; (3) Distribution: Phishers deliver emails to the victims; (4) Exploitation: Phishers exploit weaknesses of humans to lure the victims into phishing traps via socially engineered emails. (5) Exfiltration: Phishers collect sensitive data from the phishing databases.

Unlike generic phishing attacks, spear phishing targets particular individuals or organizations [31] [128] [103]. Spear phishing attacks typically extract sensitive data from their victims by attaching a type of malware to emails or in the phishing website. Industry statistics indicate that spear phishing attacks have a success rate of 19%, while the success rate of generic phishing attacks is less than 5% [103].

For the purpose of this dissertation, we will not consider email filtering (e.g., [137], [26], [85]) as a phishing detection method. Our focus is on detection of website phishing for both generic and spear phishing attacks.

(ii) Phishing Actors: There are six actors involved in a typical phishing life cycle (see Figure 3.2), as defined in the following paragraphs:

- **Phisher**: Individuals or organizations that conduct phishing attacks in order to obtain a certain type of benefit, such as financial gain, identity hiding (e.g., refers to the situation in which phishers do not use the stolen identities directly, but rather sell them to interested criminals and cyber attackers.), fame and notoriety, etc. [75][138].
- **Web service provider**: Companies that provide a certain type of service (e.g., email, social network, e-banking, on-line shopping, etc.) on the Internet (usually through a website).
- **Web service subscriber**: Customers who subscribe to web services provided by the web service provider. Subscribers are the potential targets of traditional phishing attacks.

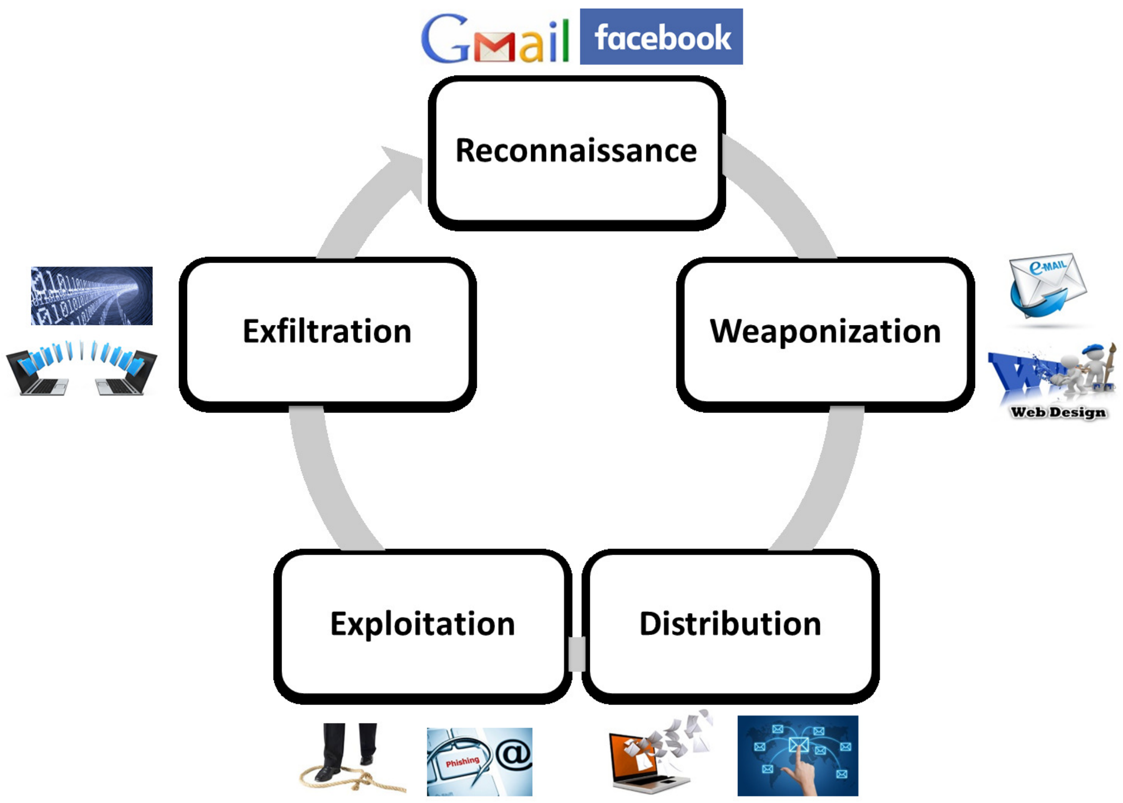


Figure 3.1 Illustration of the phishing process.

- **Web hosting provider**: Companies that provide website hosting services to web service companies.
- **Anti-phishing institutes**: Institutes that support those tackling the phishing menace and provide advice on anti-phishing controls and information on current trends [3].
- **Spear phishing targets**: Specific individuals or companies targeted by phishers.

Each actor involved in the phishing process has different actions and reactions (summarized in Table 3.3). Phishers try to use sophisticated techniques to evade phishing detection approaches (e.g., DNS poisoning [19]). In addition, there is a growing trend in which phishers have decoupled the process of phishing website hosting from the process of sending phishing emails in order to evade the anti-phishing solutions (Han, Kheir, & Balzarotti [55]).

Table 3.3 Operations of Different Players Involved in Phishing

	Basic Operations	**Advanced Operations**
A	1.Data collection 2.Website development 3.Email engineering	1. Evasion of anti-phishing techniques
B	1.Blacklist announcement	1.User behavior detection 2.Strong authentications
C	1.Human detection 2.Browser filter	1.Phishing detection toolbar
D	1.Policy enforcement	1.Brand monitoring
E	1.Phishing data analysis 2.Anti-Phishing solutions	1.Law enforcement 2.Government coalition
F	1.Employee training	1.Email filtering 2.Phishing detection software
A: Phisher; B: Web service provider; C: Web service subscriber; D: Web hosting provider; E: Anti-Phishing institute; F: Spear Phishing targets		

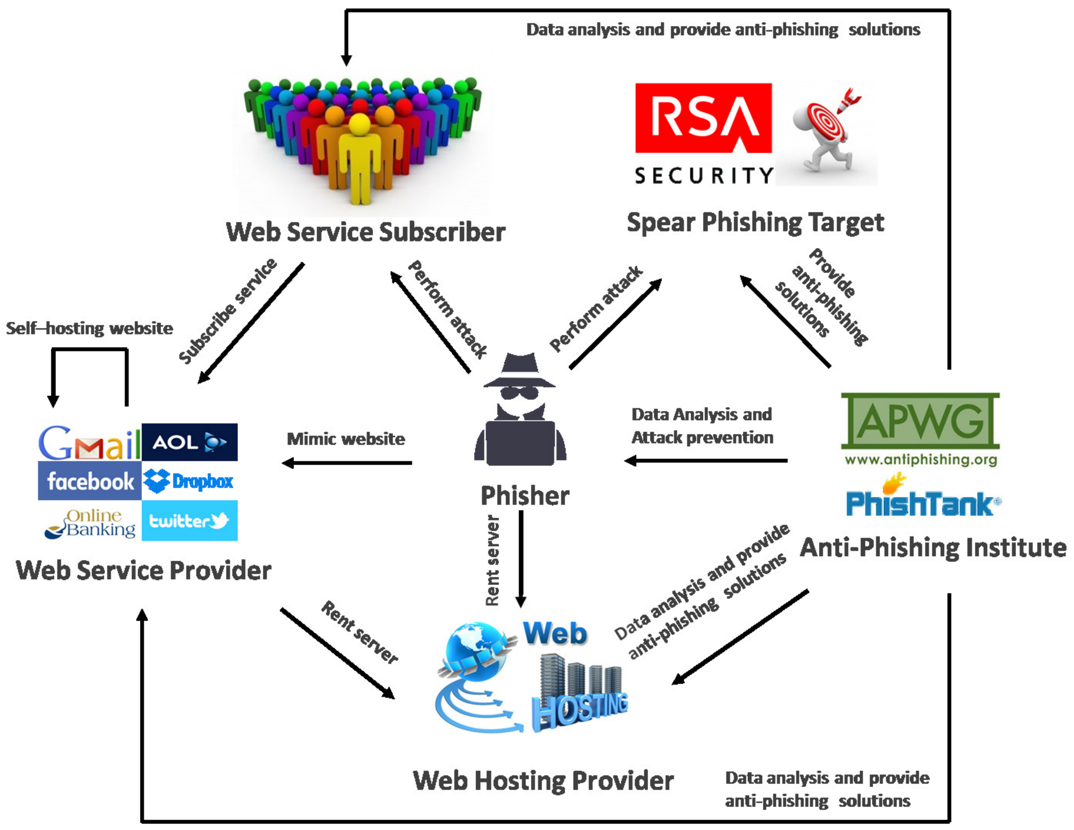


Figure 3.2 Players involved in the phishing process.

Web service providers usually announce blacklists of phishing websites and recommend users to use strong authentication schemes (e.g., [22, 70, 43, 41, 42]). Additionally, web service subscribers highly depend on browser filters (e.g., Google Safe Browser [9]) and other third party anti-phishing toolbars (e.g., Netcraft [2]) to detect and block phishing attempts.

The role of web hosting providers is rather ambiguous in the phishing process. Reputable providers usually enforce strict "Terms of Use" and avail certain anti-phishing solutions (e.g., brand monitoring [20]). Due to financial constraints, many free-to-use web hosting providers may not be able to afford deploying good anti-phishing security measures, which leaves their customers not only vulnerable, but even worse, attractive targets for phishing.

Anti-phishing institutes collect and analyze phishing data (e.g., suspicious websites reported by users) from various sources (e.g., users' reports via anti-phishing toolbars), and provide anti-phishing suggestions and solutions (e.g., up-to-date phishing website blacklist, phishing detection toolbars, etc.). In addition, they may also cooperate with government agencies such as public security and law enforcement to detect and prevent cyber attacks [3].

What is the Current State of Phishing? According to phishing activity trends reports published by APWG [23] from Jan. 2010 to Jun. 2016 (shown in Figure 3.3), the number of unique phishing websites established per month increased significantly since 2015 (i.e., the average number for 2016 is 2.93 times the average from prior years). It is clear that phishers profited from this type of cyber-attacks, which result in financial loss for both web subscribers and business owners. Therefore, agile techniques to mitigate phishing will continue to be a pressing need.

Phishing attacks tend to empoly advanced techniques to lure web service users into their rogue websites. Using the database from Trend Micro web reputation technology, Pajares [12] reports the number of phishing sites that use HTTPS connections increased significantly in 2014 compared to 2010 (shown in Figure 3.4). Attackers become more cautious and attentive when designing phishing websites to evade existing phishing detection methods [1]. Some phishing groups are capable and desire to perform more advanced phishing attacks. Avalanche (commonly known as the Avalanche Gang) is a criminal syndicate involved in phishing attacks [135]. In 2010, APWG reported that Avalanche was responsible for two-thirds of all phishing attacks in the second half of 2009, describing it as "one of the most sophisticated and damaging on the Internet" and "the world's most prolific phishing gang" [18]. It has been discovered that Avalanche uses different techniques to evade the anti-phishing mechanisms.

In addition, more and more sophisticated techniques are being used to implement phishing attacks. For example, the pharming attack, a refined version of phishing attacks, is designed to steal users' credentials by redirecting them to fraudulent websites using DNS-based techniques [47, 73]. Many computer security experts predict that the use of pharming attacks will continue to grow as more criminals embrace these techniques [65].

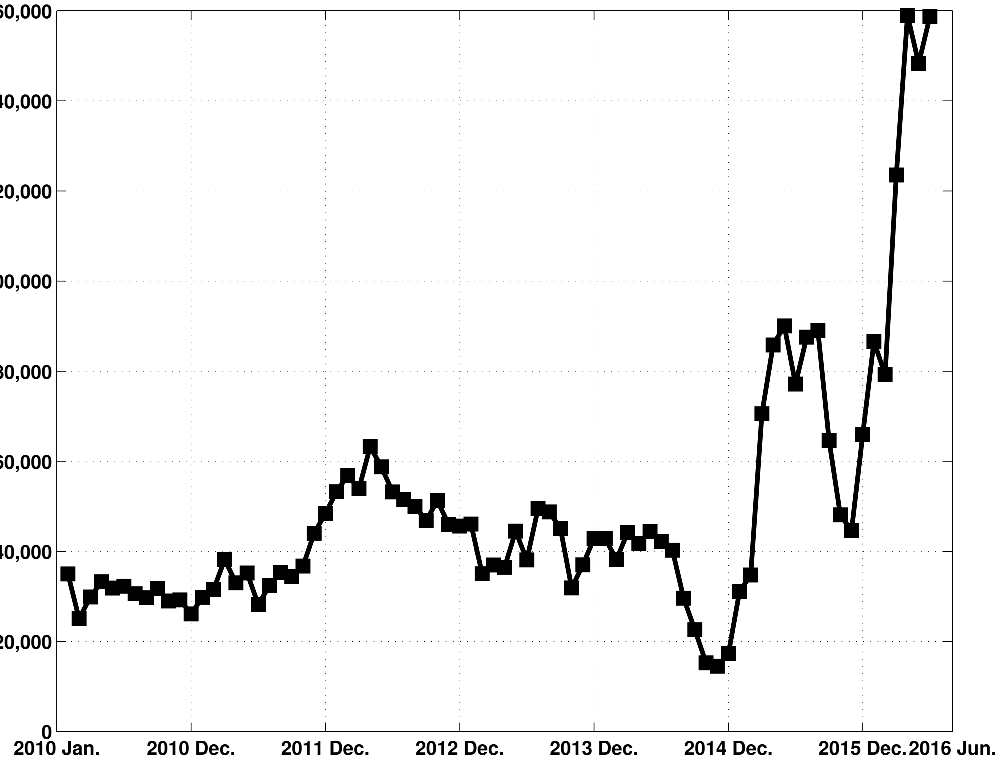


Figure 3.3 The number of unique phishing sites per month from Jan. 2010 to Jun. 2016.

3.2.2 Life Cycle of Phishing Detection

In this dissertation, we do not incorporate phishing detection approaches that rely on user education due to their poor performance. In addition, we do not cover phishing detection methods that perform email filtering because it is a different detection theme

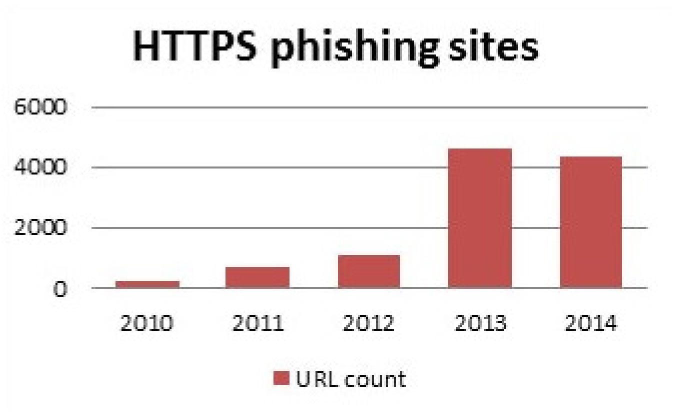


Figure 3.4 The number of phishing sites that use HTTPS. Re-printed from [12].

that warrants a separate comprehensive study on its own. we reemphasize here that our focus is on the area of software-based phishing detection which aims at detecting or blocking phishing websites.

The life-cycle of software-based phishing detection is illustrated in Figure 3.5. Starting from the initial inputs, the detection scheme extracts phishing detection features (or called heuristics) and/or blacklists from various sources (e.g., URL related information, trusted third party, WHOIS server, etc.) via different feature mining approaches (e.g., search engines, target identification algorithms, etc.). Then, it applies different data mining algorithms and/or proposes various detection strategies to the engineered features to achieve its objectives (e.g., identifying phishing links, blocking phishing websites, etc.). To evaluate the performance of phishing detection schemes, various evaluation datasets are collected from different sources (e.g., PhishTank, Yahoo directory, etc.). Finally, leveraging the collected datasets and following various validation strategies (e.g., cross validation), the proposed scheme is evaluated based on multiple metrics (e.g., False Positive Rate, False Negative Rate, etc.).

In the coming sections, following the life cycle of software-based phishing detection schemes, we present a comprehensive study of the phishing detection research from five different perspectives, namely, classification of phishing detection techniques, validation datasets, detection features, detection techniques and detection criteria.

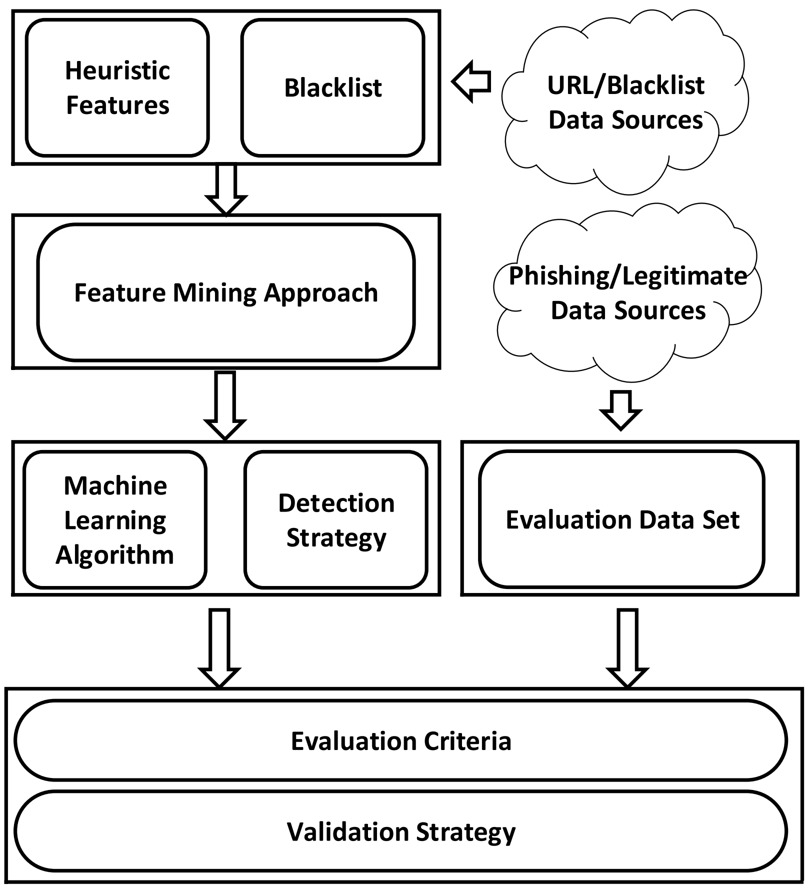


Figure 3.5 Life Cycle of typical phishing detection schemes.

3.3 Phishing Detection: Feature Analysis and Selection

In this section, we summarize the most commonly used features taken from various Phishing detection approaches. Then, we introduce a new phishing detection feature based on $NRTT$. Finally, based on the analysis of the presented features, we perform careful selection from these features to build a new Phishing detection framework. We note that even though the list of atomic features presented here is not exhaustive, it includes the atomic features used in most of the state-of-the-art Phishing detection approaches.

3.3.1 Atomic Phishing Detection Features

URL-based textual features: A URL (Uniform Resource Locator) is used to locate website resources. URLs contain many features that have been widely used in various Phishing detection approaches [28] [136] [133], here are some most used features:

- **URL replaced with IP address**: Some Phishing websites do not use host-names, but rather use IP address directly to locate the fake site. Such behavior usually used either to obfuscate the legitimate URL or simply to reduce cost.
- **The Length of URL**: Phishing websites usually have longer URLs comparing to legitimate websites.
- **Number of dots and sub domains**: Phishing URLs often contain more "dots" and sub-domains than legitimate.
- **Number of re-directions**: Malicious URLs often have multiple URL redirects in order to evade detection by blacklists.
- **Use of HTTPS protocol**: Legitimate websites often use HTTPS protocol while Phishing sites usually do not.

URL-based domain name features:

- **Domain name similarity**: A measure of the similarity between a potential Phishing domain name and a target domain name. The similarity can be measured in many ways. For example, it can be measured based on the Edit Distance between the two domains [36]. The Edit Distance is the number of characters that need to be inserted or deleted in order to transform one domain into another. The smaller the number of insertions and deletions, the higher the similarity.

WHOIS information: WHOIS is a query and response protocol that is widely used for querying databases that store information about registered websites [89].

- **Registered information about domain names**: For most of the observed Phishing sites, either the registration record is not available in WHOIS databases or the claimed identity is not accurate in the record.
- **Age of Domain**: Many of the observed Phishing websites have domains that are registered only a few days before Phishing emails are sent out, that is, short lived domains are likely to be Phishing domains.

Geographic information: Geographical location is one of the most commonly used indicators in detecting Phishing because Phishing websites are likely to be hosted in locations different from those of legitimate websites [2].

- **IP-based Geographical location**: Netcraft [2] provides location information (i.e., IP-based country information) to help identifying fraudulent URLs. For example, the real bankofamerica.com is unlikely to be hosted in Russia.

Website textual content: The textual content of the Phishing website can be used to determine the identity of the target website.

- **Term frequency**: In [136], the author proposed a content-based approach to detect Phishing websites based on the TF-IDF (term frequency, inverse document frequency) information with the help of Google search engine.
  - TF-IDF of each term on a suspected web page is calculated.
  - Top 5 terms with highest TF-IDF values are selected.
  - The top 5 terms are submitted into a search engine and store the domain names of the n-first returned entries.
  - If the suspected domain name is found within the n returned results, then the site is considered legitimate.

Website visual content:

- **Visual similarity**: In [91], the authors present a technique to visually compare a suspected Phishing page with the legitimate one via a set of visual features. These features include (i) each visible text section with its visual attributes, (ii) each visible image, and (iii) the overall visual look-and-feel (i.e., the larger composed image) of the web page visible in the viewport (i.e., the part of the web page that is visible in the browser window).

3.3.2 Network Round Trip Time

The usage of $NRTT$ as a feature for Phishing detection is straightforward. We simply compare the differences between the $NRTT$ statistics (i.e., mean and standard deviation) of the phishing website and that of the its targeting website based on the algorithms discussed in Section 3.4.

Gaussian approximation: It has been observed that the round trip network communications latency approximately follows a Gaussian distribution [37] [61]. Based on this assumption, the larger the number of profiling signals, the more accurate the profile. However, the larger the number of profiling signals, the higher the bandwidth overhead consumed by the verifier. Therefore, it is critical to find a profiling sample size that leads to an acceptable trade-off between profile accuracy and bandwidth overhead.

According to the work in [61], for a Gaussian distribution that has standard deviation σ and mean μ. The minimum sample size N, that produces a mean x, within a certain error tolerance δ, with a certain confidence level $1 - \alpha$, can be computed as:

$$N \geq (Z_{1-\alpha}/\delta)^2\sigma^2 \tag{3.1}$$

Where Z is the critical value for the normal distribution. In other words, for a sample size of N, we are 1-α confident that the measured mean (x) will fall in the range of:

$$\mu - \delta \leq x \leq \mu + \delta \tag{3.2}$$

For example, to be 99% confident that x is in the range of $\pm\delta = \pm 0.5 \cdot \sigma$ (i.e., the error tolerance is half of the standard deviation), the number of measurements should be: $N \geq (Z_{1-\alpha}/\delta)^2\sigma^2 = 4 \cdot Z_{0.99} \approx 27$. In other words, with more than 27 profiling signals, we are 99% confident that real-time profile mean is within $0.5 \cdot \sigma$ from reference profile mean.

Network instabilities: *NRTT* measurements should be robust enough to resist manipulation attacks and should cope with network instabilities to prevent high false positive. These *NRTT* challenges has been well studied and address in [61], where the authors define three types of network instabilities provide solution that efficiently address each type of these instabilities.

Instantaneous instabilities cause transient changes in *NRTT* and hence, they only affects a few of the profiling signals. This type of instability can be addressed through outlier filtering based on median absolute deviation[84] .

Long-term instabilities are instabilities that persist long enough to affect all or most of the of profiling signals yet not permanent. These instabilities are mainly caused by traffic congestion at the local network segment connecting to the network backbone. This type of instability can be addressed by establishing multiple profiles as detailed in [61].

Routing instabilities will result in permanent changes in network communications latency due to, for example, permanent network routing changes. It has been shown in many previous researches [38] [81] [109] [82] [114], that only a small portion of the Internet is responsible for the vast majority of the routing instabilities and these routing changes exhibit a strong temporal periodicity, despite the growth of the Internet. We can leverage such periodicity and temporal properties to create long-term dynamic temporal profiles.

System design for *NRTT* measurement: We propose a system prototype (Figure 3.6) to turn *NRTT* into a robust feature for the proposed phishing detection framework. The prototype consists of four entities: (1) the user (represented by a plug-in); (2) the verifier; (3) the locating servers; and (4) the website server. The verifier and the locating servers form a phishing detection network which are third trusted parties. This is a valid design because most of the client side phishing detection toolbar providers host their own servers that interact with the user's browser plug-in (e.g., blacklist updating, user feedback uploading, etc.)

Using the proposed system, the high level measurement procedure involves: (1) the user installs a browser plug-in and active it; (2) the user opens the faked website link; (3) the plug-in collects the URL of the faked website and sends it to the verifier; (4) the verifier goes to the faked website and finds out the target website (e.g., PayPal) using the algorithm described in Section 3.4.2; (5) the verifier sends testing requests to multiple Locating Servers (LS) to measure the $NRTT$ from each LS to the URL hosting server; (6) the verifier leverages the $NRTT$ measurements together with the proposed phishing detection framework (as detailed in Section 3.4) to identify potential phishing websites.

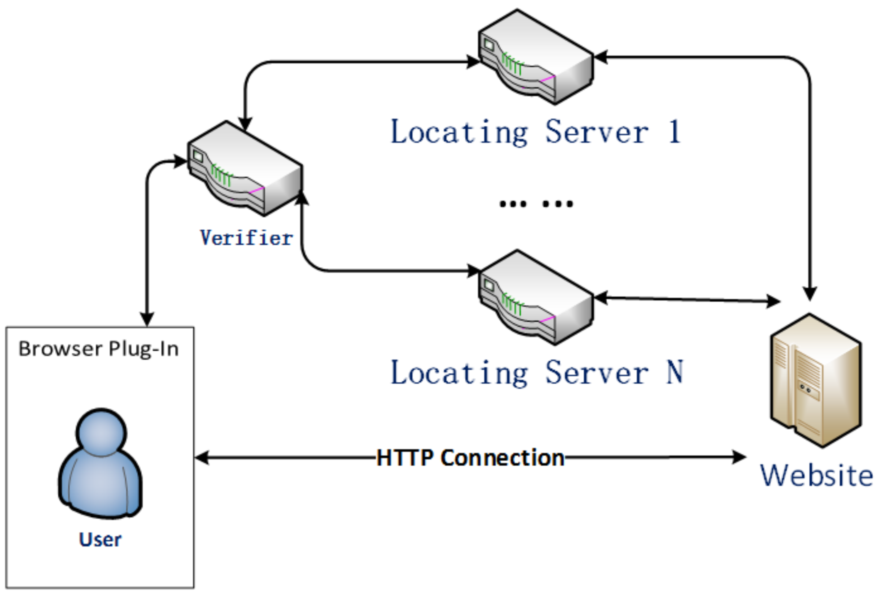


Figure 3.6 System model for the measurement of $NRTT$.

3.3.3 Selection of the Features

In this section, we perform the analysis and selection of the phishing detection features for the proposed framework. First of all, we classify all the features into two categories, namely, Identification Dependent Features (i.e., IDF) and Identification

Independent Features (i.e., IIF). Together with the proposed target identification algorithm (Section 3.4.2), IDF could achieve better detection accuracy. For example, in [36], the proposed phishing detection toolbar SpoofGuard leverages the domain name similarity score. However, since it does not provide targeting identification of the phishing website, it simply compares the phishing domain name with the the domains in the most recent browser history entries. And this results in bad performance.

The URL-based features and WHOIS features are still powerful in classifying the phishing websites due to the attackers' laziness and their limited resources. However, webpage content based features suffer from more difficulties than usual [98]. For example, the attacker can encrypt the phishing website source code to evade website content based phishing detection methods. Geographic location is a very important phishing detection feature because the phishing website is most likely hosted in a place other than the one of the legitimate website [2]. The usage of $NRTT$ provides similar ability comparing to the IP based geo-location. In addition, it provides better detection accuracy (e.g., current IP based geo-location schemes only provide country/state level information) and it could defend against IP/BGP hijacking, which is even more common than researchers think according to the work in [127]. The selected phishing detection features are summarized in Table 3.4.

3.4 Phishing Detection Framework Design

3.4.1 Design Overview

In this section, we present the system design of the proposed Phishing detection framework. As depicted in Figure 3.7, the proposed framework comprises three modules: (1) Phishing target identification module (TIM); (2) identification dependent phishing detection module (IDDM); and (3) identification independent phishing

Table 3.4 Selected Phishing Detection Features for the Proposed Framework

Set	**Feature**	**Symbol**
IDF	Network Communications Round Trip Time	$NRTT$
	Domain Name Similarity	DNS
IIF	# of dots in the URL	N_{dots}
	IP address in the URL	IP_{url}
	# of redirections of the URL	N_{re}
	Use of HTTPS	P
	Age of the Domain	A
	Registering name in WHOIS	R_{domain}
	Length of the URL	L_{url}

detection module (IIDM). The detailed Phishing detection procedure is presented in Algorithm 1.

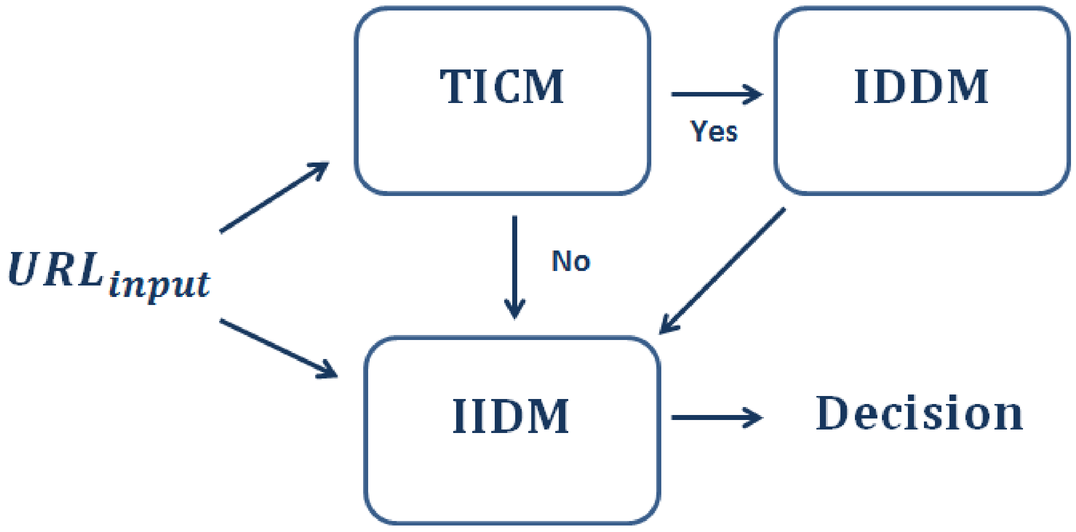


Figure 3.7 Design overview.

- **TIM**: the Phishing target identification and classification module is responsible of identifying the target website from input URL. The identified result is in the form of domain names (e.g., *www.paypal.com*) and it is the premise of the identification dependent phishing detection (IDDM). Detailed design and analysis of TIM are presented in Section 3.4.2.
- **IDDM**: the identification dependent phishing detection module performs the detection algorithm for all the features that require the target identification of the website (i.e., Set IDF). Detailed design and analysis of IDDM is presented in Section 3.4.3.
- **IIDM**: If TIM cannot verify the identification of the website (less than 5% of according to the experimental results), IIDM will take over the control and perform phishing detection based on website identification independent features (i.e., Set IIF). On the other hand, if TIM successfully identifies the target website, IDDM will first handle the identification dependent features then forward the results (i.e., $score_{IDDM}$) to IIDM for final decision. Detailed design and analysis of IIDM are presented in Section 3.4.4

Algorithm 1 Phishing Detection Procedure

```
    Input: $URL_{input}$
    Output: Phishing detection decision: True or False
 1: procedure
 2:     Send $URL_{input}$ to TIM, $D_l$= call Algorithm 2
 3:     if TIM successfully identifies the target then
 4:         TIM sends the $D_l$ to IDDM.
 5:         IDDM performs Algorithm 3
 6:         IDDM forwards $score_{IDDM}$ to IIDM
 7:         IIDM performs Algorithm
 8:         if $score_{IIDM} \geq threshold$ then
 9:             return False
10:         else
11:             return True
12:         end if
13:     else
14:         TIM forwards the control to IDDM.
15:         IIDM performs Algorithm
16:         if $score_{IIDM} \geq threshold$ then
17:             return False
18:         else
19:             return True
20:         end if
21:     end if
22: end procedure
```

3.4.2 Target Identification

Identification process: The proposed website target identification and classification module (TIM) aims at finding out the identification, in terms of target domain name (e.g.,*www.paypal.com*), based on the input URL. In this work, TIM first extracts a list of keywords from several selected website contents. The selected website contents include:

- URL_{input}: the input URL.
- T_{title}: the title of the website.
- $T_{copyright}$: the copyright field of the website.
- T_{text}: other website text contents, including: pure text, text from the hyperlink, text extracted from the images.

After obtaining the list of keywords, TIM tries to match the list of keywords to a pre-defined web service brands whitelist which is further mapped to its corresponding domain name (e.g, Paypal $\rightarrow$ *www.paypal.com*). This whitelist includes 92 most phished websites information which are collected from 37321 verified phishing websites of PhishTank [100]. If the extracted list of keywords has no match to the whitelist, TIM will apply the list of keywords to a searching engine (i.e., Google search) and record the domain names of the top N (e.g., $N = 3$) searching results.

As depicted in Figure 3.8 and **Algorithm** 2., the website target identification and classification module (TIM) consists of 6 sub-modules:(1) Module 1 simply records the input URL as a string in $SetA$; (2) Module 2 extracts the page title T_{title} and copyright field $T_{copyright}$ from the HTML source code then records them as $SetB$; (3) Module 3 calculates the term frequency of the all other web page texts T_{text}, then saves the results as $SetC$ (i.e., from text only) and $SetD$ (i.e., from the image extracted text); (4) Module 4 performs the whitelist matching among all the sets (i.e, $SetA, B, C, D$), and saves the hitting list as $ListA$. (5) Module 5: if there is no matching resolved from Module 4, Module 5 will construct $ListB$ in the form of $(SetB, SetC, SetD)$; (6) Module 6 maps $ListA$ or $ListB$ into corresponding domain names based on the whitelist or the top N $(N = 3)$ searching result, respectively.

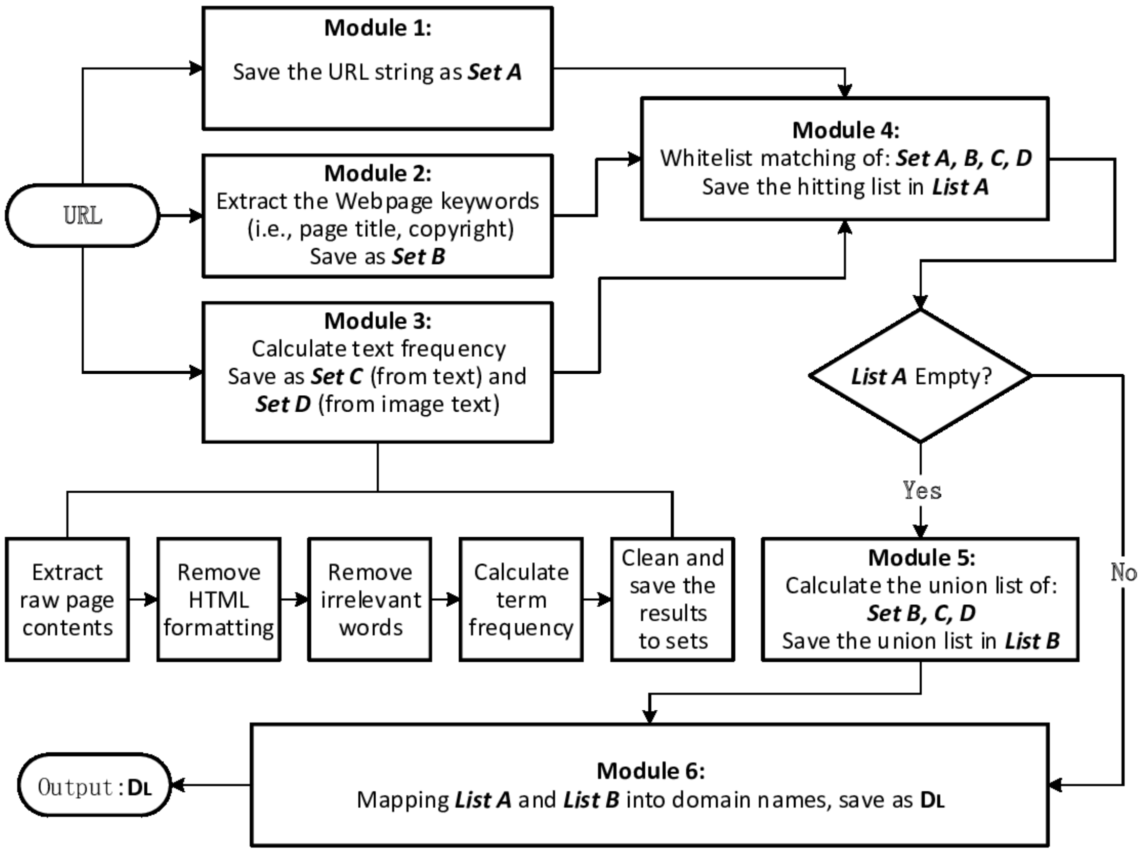

URL
Module 1:
Save the URL string as Set A
Module 2:
Extract the Webpage keywords
(i.e., page title, copyright)
Save as Set B
Module 3:
Calculate text frequency
Save as Set C (from text) and
Set D (from image text)
Module 4:
Whitelist matching of: Set A, B, C, D
Save the hitting list in List A
List A Empty?
Yes
No
Extract raw page contents
Remove HTML formatting
Remove irrelevant words
Calculate term frequency
Clean and save the results to sets
Module 5:
Calculate the union list of:
Set B, C, D
Save the union list in List B
Module 6:
Mapping List A and List B into domain names, save as DL
Output: DL

Figure 3.8 TIM working flow.

Special Cases: We note that there are special scenarios that TIM cannot generate any outputs (i.e., $D_l = \varnothing$), for example, the website source code is protected via encryption and there is no image on the website. Under such scenario, TIM will fail to provide outputs and IIDM will take over the control of the detection process as depicted in Figure 3.7. Alternatively, we can take the screen shoot of the webpage for further analysis. However, it will increase the system complexity and detection time overhead. In addition, according to our investigation of 500 legitimate websites,

Algorithm 2 Target Identification Algorithm

```
    Input: $URL_{input}$
    Output: Target domain name $D_l$
 1: procedure
 2:     Feed $URL_{input}$ to Module 1 to 3
 3:     Module 1 to 3 outputs $SetA, B, C, D$.
 4:     Module 4 matches $SetA, B, C, D$ with the whitelist.
 5:     if There are matches in Module 4 then
 6:         Record the matches in $ListA$
            Module 6 maps $listA$ into corresponding domain
            names and records the result as $D_l$.
 7:     else
 8:         Create $ListB$ in the form of $(SetB, SetC, SetD)$.
            Module 6 applies $ListB$ to the searching engine
            and records the top $N$ domain names as $D_l$.
 9:     end if
10:     Return: $D_l$
11: end procedure
```

encryption is not used by any of these legitimate web services and we can confidently conclude a phishing. Another possible scenario is that we disable the text extraction from the website images in order to reduce the overall detection overhead.

3.4.3 Identification Dependent Phishing Detection Module

After obtaining the target identification output (i.e., D_l) from TIM, IDDM takes over the control of the detection process. IDDM performs evaluation of two target identification dependent features, namely, normalized domain name similarity score and the $NRTT$ Gaussian PDF score. After the evaluation, IDDM will forward the results to IIDM for final analysis. Detailed evaluation process is presented in **Algorithm 3**.

Normalized Domain Name Similarity: First, IDDM extracts the domain name embedded in the URL_{input} and records it as string a_i with length $|a|$. Then, it denotes D_l as string b_j with length $|b|$. Finally, IDDM calculates the normalized domain name similarity score ($score_{DNS}$) using equation (1) and (2) based on the edit distance of the two strings.

$$D_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0 \\ min \begin{cases} D_{a,b}(i-1,j)+1 \\ D_{a,b}(i,j-1)+1 \\ D_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases} \tag{1}$$

$$score_{DNS} = \frac{max(|a|,|b|) - D_{a,b}(|a|,|b|)}{max(|a|,|b|)} \tag{2}$$

***NRTT* Gaussian PDF score:** As detailed in Section 3.3.2, the IDDM builds m corresponding real-time profiles for the website represented by URL_{input}, denoted as $U(\mu, \sigma)$. Then, IDDM retrieves the pre-stored $NRTT$ profiles (i.e., to reduce usage of the bandwidth) or measures the real time $NRTT$ profiles (i.e., to alleviate the effect of $NRTT$ instability or in the case of no pre-stored reference profiles) for the websites in D_l, denoted as $V(x, y)$. Finally, IDDM computes the distance between $U(\mu, \sigma)$ and $V(x, y)$. To measure the distance, we use the Gaussian PDF algorithm [123] as shown in equation (3):

$$score_{NRTT} = \frac{1}{m} \sum_{i=1}^{i=m} e^{-\frac{(x_i - \mu)^2}{2 \cdot \sigma^2}} \tag{3}$$

We note that, there may be multiple domain names in D_l because of the uncertainty of TIM (e.g., keywords matching, top N searching). Under this scenario, IDDM simply calculates $score_{DNS}$ and $score_{NRTT}$ for each website in D_l and keeps the highest score respectively. This operation helps to decrease the false positive rate of the phishing detection system.

3.4.4 Identification Independent Phishing Detection Module

The IIDM performs the evaluations of all the target identification independent features (i.e., Set IIF). It computes the corresponding evaluation score for each of the features in Set IIF. The corresponding evaluation score calculations are presented in Table 3.5. Finally, IIDM calculates the final score $score_f$ based on equation (11). The decision algorithm is presented in **Algorithm** 3.5.

There are three types of evaluation scores computed in the proposed framework:

- **Comparative score**: $CS = score_{NRTT} + score_{DNS}$. CS depends on the outputs of the IDDM that requires the target identification of the input URL.
- **WHOIS score**: $WS = score_6 \cdot score_7$. For the website being examined, if the there is no registering information in the WHOIS record (i.e., $score_6 = 0$), it is most likely to be a phishing site and we can confidently set $WS = 0$.
- **URL score**: $US = score_1 \cdot \sum_{i=2}^{i=5}(score_i)$. For the website being tested, if the corresponding URL contains IP address, it is most likely to be a phishing site and we can confidently set $US = 0$.

Algorithm 3 Identification-Dependent Feature Evaluation

```
    Input: $a_i$ = Domain name of $URL_{input}$)
 1:        $b_i = D_l$
    Output: $score_{DNS}$ and $score_{NRTT}$
 2: procedure
 3:     $D_{a,b}(i,0) = 0,\ D_{a,b}(0,j) = 0$
 4:     for $i \in [1, 2, ..., |b|]$ do
 5:         for $j \in [1, 2, ..., |a|]$ do
 6:             $D_{a,b}(i,j) =$ Equation (1)
 7:         end for
 8:     end for
 9:     $score_{DNS} = \frac{max(|a|,|b|) - D_{a,b}(|a|,|b|)}{max(|a|,|b|)}$
10:     $score_{NRTT} = 0$
11:     for $j \in [1, 2, ..., m]$ do
12:         $x_j = x_j - \triangle T$
13:         $score_{NRTT} += e^{-\frac{(x_j - \mu)^2}{2 \cdot \sigma_j^2}}$
14:     end for
15:     $score_{NRTT} = score_{NRTT}/m$
16:     Return: $score_{DNS}$ and $score_{NRTT}$
17: end procedure
```

Table 3.5 Evaluation Score Calculation Formulas for Identification Independent Features

Features	**Calculation Formula**
IP_{url}	$score_1 = \begin{cases} 0 & \text{if } URL_{input} \text{ contains IP} \\ 1 & otherwise \end{cases}$ (4)
L_{url}	$score_2 = \begin{cases} 1 & \text{if } L_{url} <= L_{threshold} \\ L_{threshold}/L_{url} & otherwise \end{cases}$ (5)
N_{dots}	$score_3 = \begin{cases} 1 & \text{if } N_{dots} <= N_{threshold} \\ N_{threshold}/N_{url} & otherwise \end{cases}$ (6)
N_{re}	$score_4 = \begin{cases} 1 & \text{if } N_{re} <= NR_{threshold} \\ NR_{threshold}/N_{re} & otherwise \end{cases}$ (7)
P	$score_5 = \begin{cases} 1 & \text{if HTTPS is used} \\ 0 & otherwise \end{cases}$ (8)
A	$score_6 = \begin{cases} 1 & \text{if } A <= A_{threshold} \\ A_{threshold}/A & otherwise \end{cases}$ (9)
R_{domain}	$score_7 = \begin{cases} 1 & \text{if } R_{domain} \text{ exists} \\ 0 & otherwise \end{cases}$ (10)

The final score is calculated as:

$$score_f = \frac{w_1 \cdot CS + w_2 \cdot WS + w_3 \cdot US}{w_1 + w_2 + w_3} \tag{11}$$

Algorithm 3 Decision Algorithm

```
    Input: $URL_{input}$, $score_{DNS}$ and $score_{NRTT}$
    Output: Detection decision: True or False
 1: procedure
 2:     Compute $CS = score_{NRTT} + score_{DNS}$
 3:     Compute $WS = score_6 \cdot score_7$, where
 4:             $score_6$ = Call equation (9)
 5:             $score_7$ = Call equation (10)
 6:     Compute $US = score_1 \cdot \sum_{i=2}^{i=5}(score_i)$, where
 7:             $score_2$ = Call equation (5)
 8:             $score_3$ = Call equation (6)
 9:             $score_4$ = Call equation (7)
10:             $score_5$ = Call equation (8)
11:     Calculate the final score:
12:             $score_f = \frac{w_1 \cdot CS + w_2 \cdot WS + w_3 \cdot US}{w_1 + w_2 + w_3}$
13:     if $score_f \geq threshold$ then
14:         return True ← Legitimate
15:     else
16:         return False ← Phishing
17:     end if
18: end procedure
```

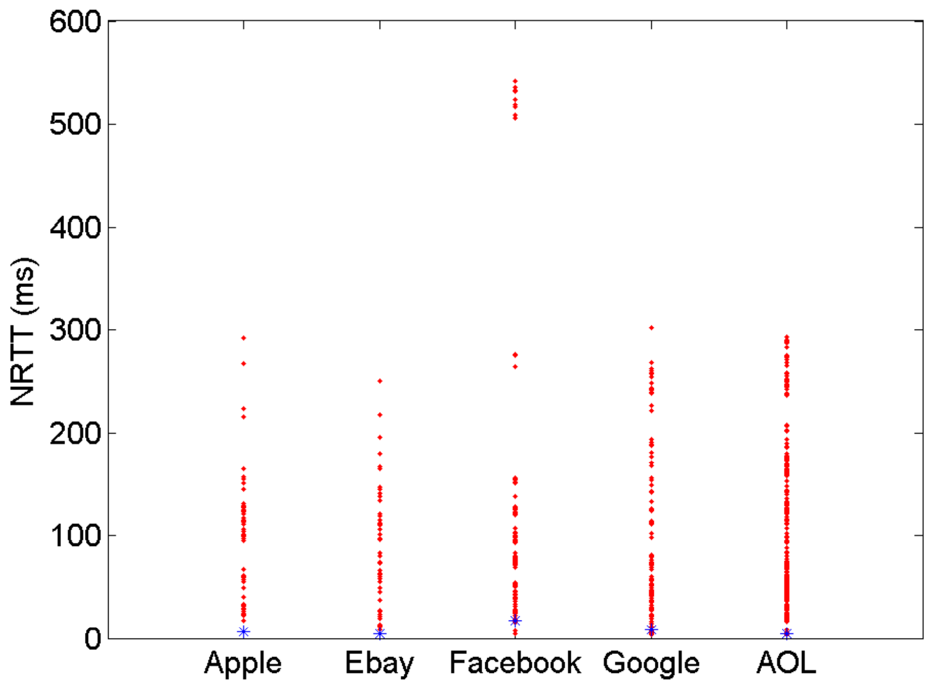


Figure 3.9 $NRTT$ distributions for five popular web services brands; Blue dots $\rightarrow$ Legitimate; Red dots $\rightarrow$ Phishing.

3.5 Performance Evaluation

In this section, we present the performance evaluation of the selected and proposed phishing detection features, the target identification method and the proposed phishing detection framework.

3.5.1 Evaluation Data Sets

We obtain the evaluation data sets mainly from three sources: (1) 2000 to 32683 verified phishing websites provided by PhishTank [100]; (2) Top 500 most visited legitimate websites list from Alexa (i.e., *www.alexa.com*); and (3) 142 the manually collected legitimate websites from other Internet resources. The detail about the data sets used for each experiment is presented in the following corresponding subsections.

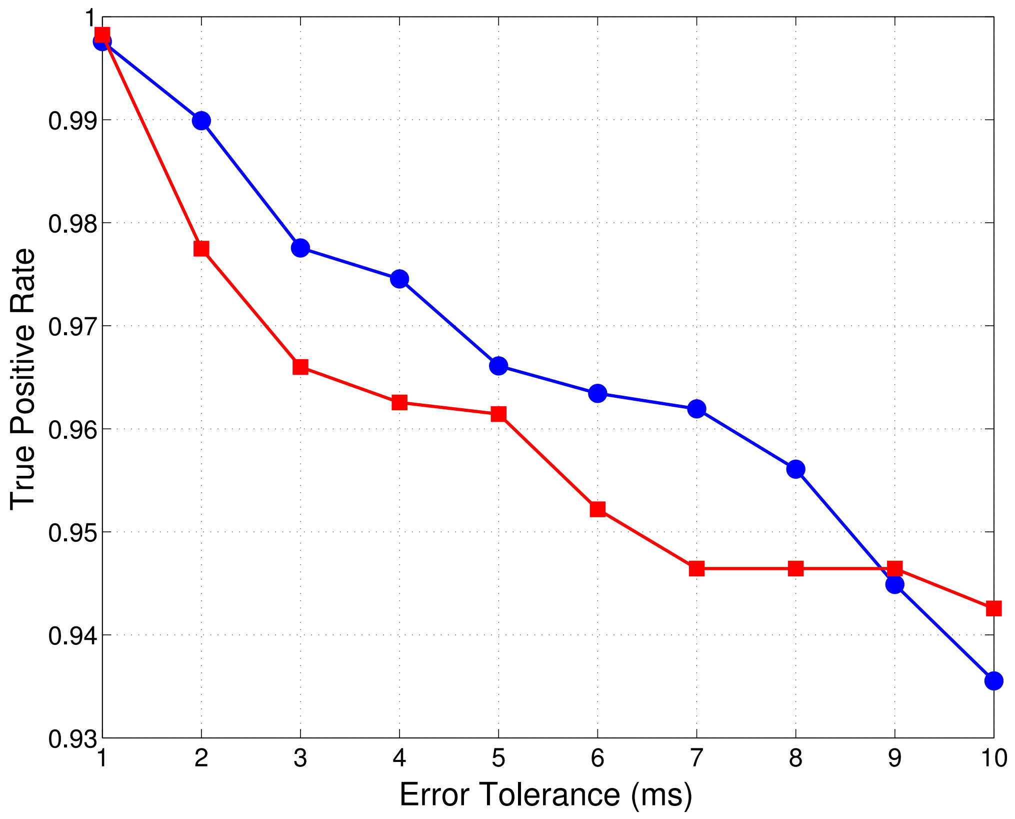


Figure 3.10 True Positive Rate using $NRTT$ for various error tolerance values.

3.5.2 Evaluation of Features and Feature Sets

In this subsection, we evaluate the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) among different selected features and different feature sets. TPR measures the proportion of positives that are correctly identified (i.e., the percentage of phishing sites who are correctly identified):

$$TPR = \frac{Number \quad of \quad correctly \quad identified \quad sites}{Total \quad number \quad of \quad phishing \quad sites}$$

FPR measures the proportion of positives that are wrongly identified (i.e., the percentage of legitimate sites who are wrongly identified as phishing sites):

$$FPR = \frac{Number \quad of \quad wrongly \quad identified \quad sites}{Total \quad number \quad of \quad legitimate \quad sites}$$

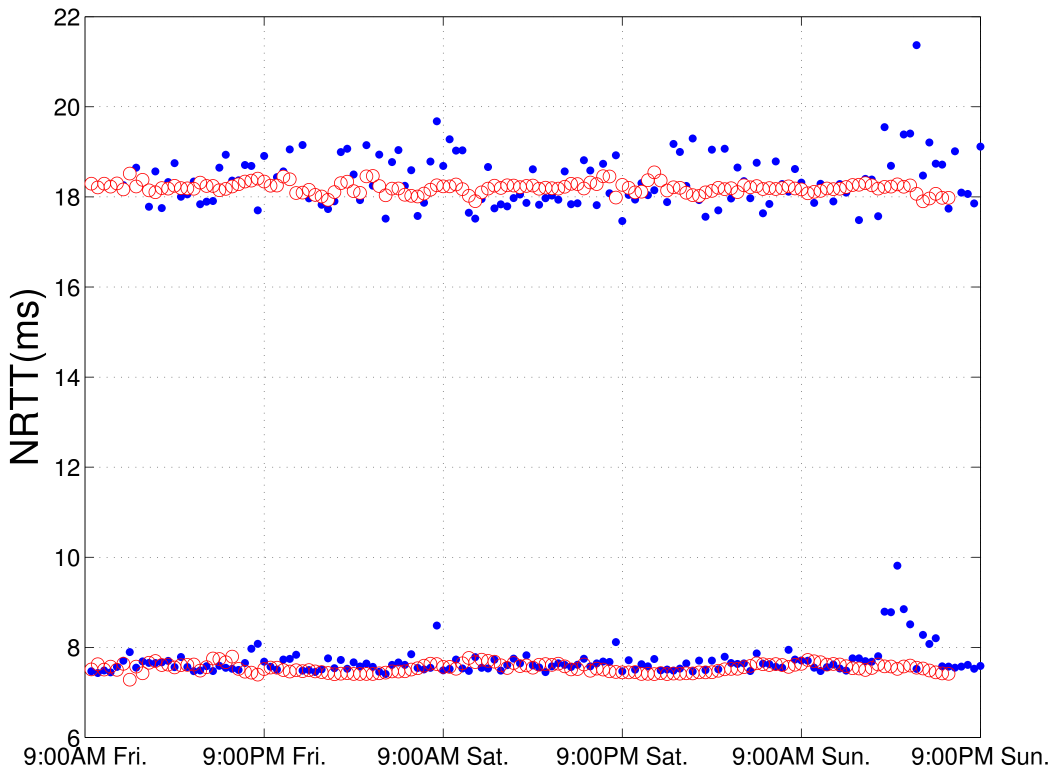


Figure 3.11 Alleviation of the $NRTT$ instabilities.

Figure 3.16 shows FPR and TPR for all the binary features, including: IP_{url},P and R_{domain}. We can conclude that IP_{url} and R_{domain} introduce 0 FPR while they can only help to classify $< 20\%$ phishing websites which are poorly designed. For P, as more and more phishing websites deploy the usage of HTTPS protocol, its contribution is getting lower and lower. However, it still helps in current state. Figure 3.12 shows the ROC (Receiver operating characteristic) curve of FPR vs. TPR for different URL based features includes L_{url}, N_{dots}, N_{re} and the URL set (i.e., combining of all the 3 features plus the binary features P and IP_{url}). It shows that the URL set alone cloud achieve about 90% TPR with 2% FPR. Figure 3.13 shows the ROC curve of FPR vs. TPR for different feature sets, including: WHOIS set, URL set, web of trust score and the combination of all the 3 sets. The WHOIS set contains 2 features, namely,

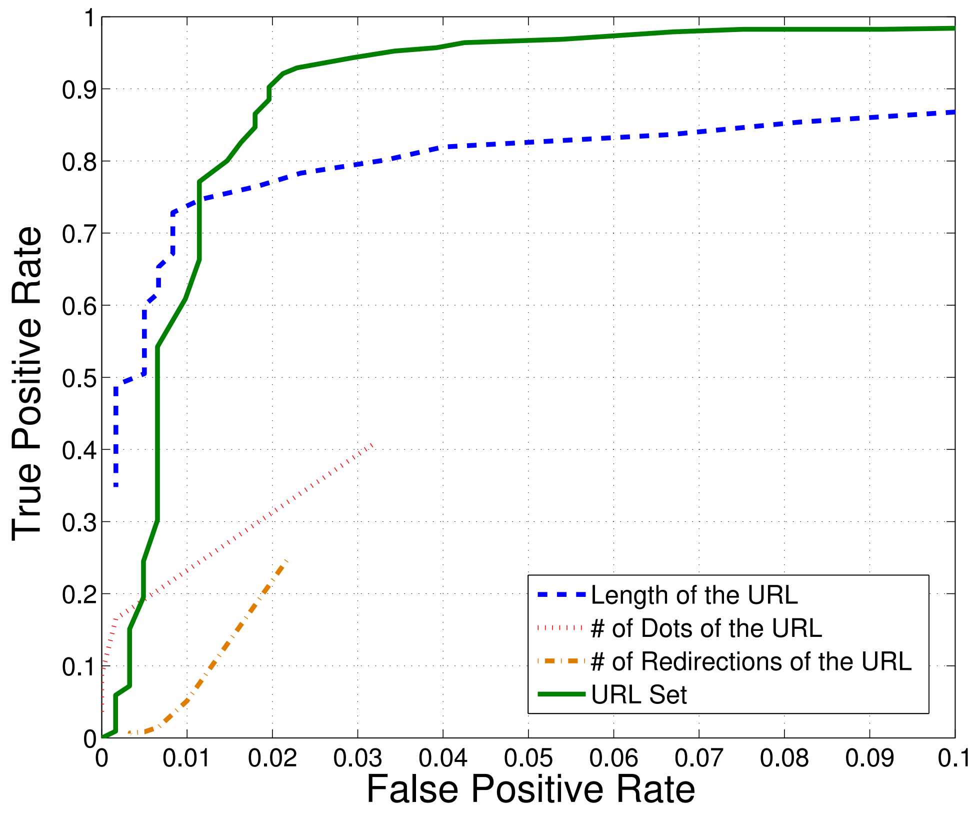


Figure 3.12 ROC curves of FPR-TPR for different URL based features and URL set.

the age of the domain and the existence of the registering information in WHOIS database. The web of trust score is provided by SEO (search engine optimization) that collects all website ranking information based on Google, Bing, Yahoo etc. The resutls shows the combination of all the selected feature sets can achieve about 93% TPR with 0.5% FPR. Figure 3.14 shows the ROC curve of FPR vs. TPR for all the three selected feature set, $NRTT$ and all the feature set (i.e., the proposed scheme). It clearly shows that with the combination of all the features, the proposed scheme can achieve 99% TPR and 0.2% FPR.

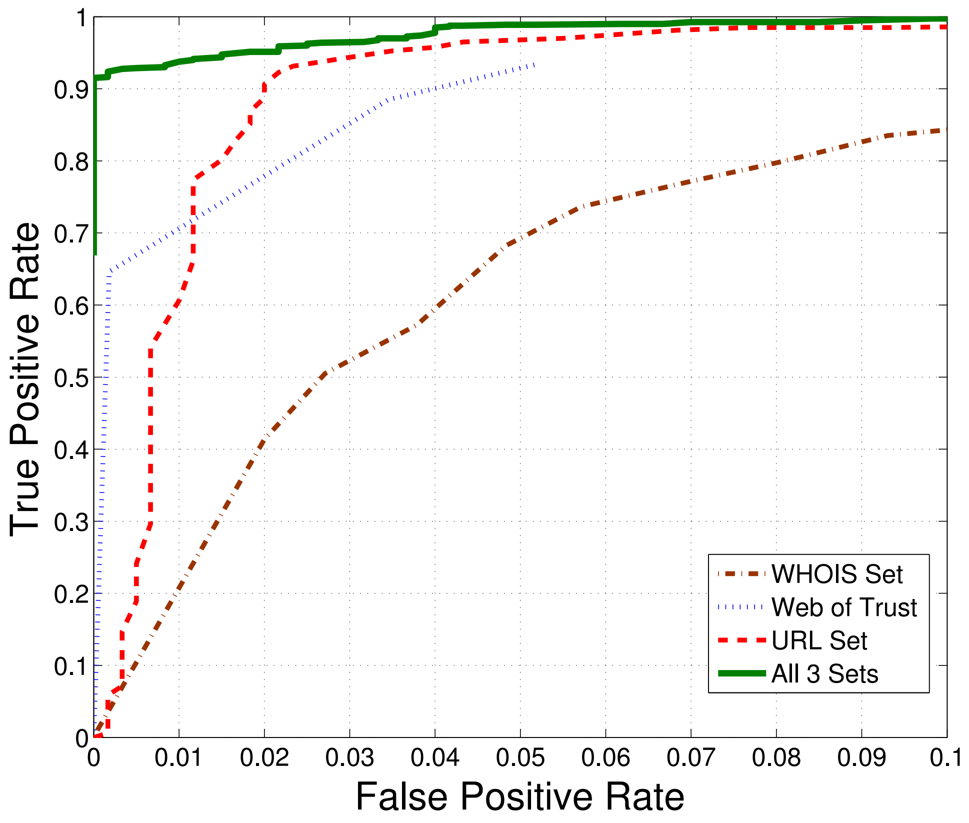


Figure 3.13 ROC curves of FPR-TPR for different feature sets.

3.5.3 Evaluation of $NRTT$

This evaluation aims to prove that $NRTT$ can achieve high detection accuracy in classifying verified on-line phishing websites. In the first test, we collect the $NRTT$ data (i.e., measured from a fixed server to all the destinations) of 928 verified on line phishing websites that are targeting five popular web service brands, namely, Apple, Ebay, Facebook, Google and AOL. Figure 3.9 shows the corresponding phishing websites $NRTT$ distributions for each web service brand. The red dots denote the phishing website's $NRTTs$ while the blue dots are the $NRTT$ of the legitimate websites. We can see that the legitimate websites tend to have smaller $NRTT$ values while the phishing websites have various $NRTT$ values ranged from a few million

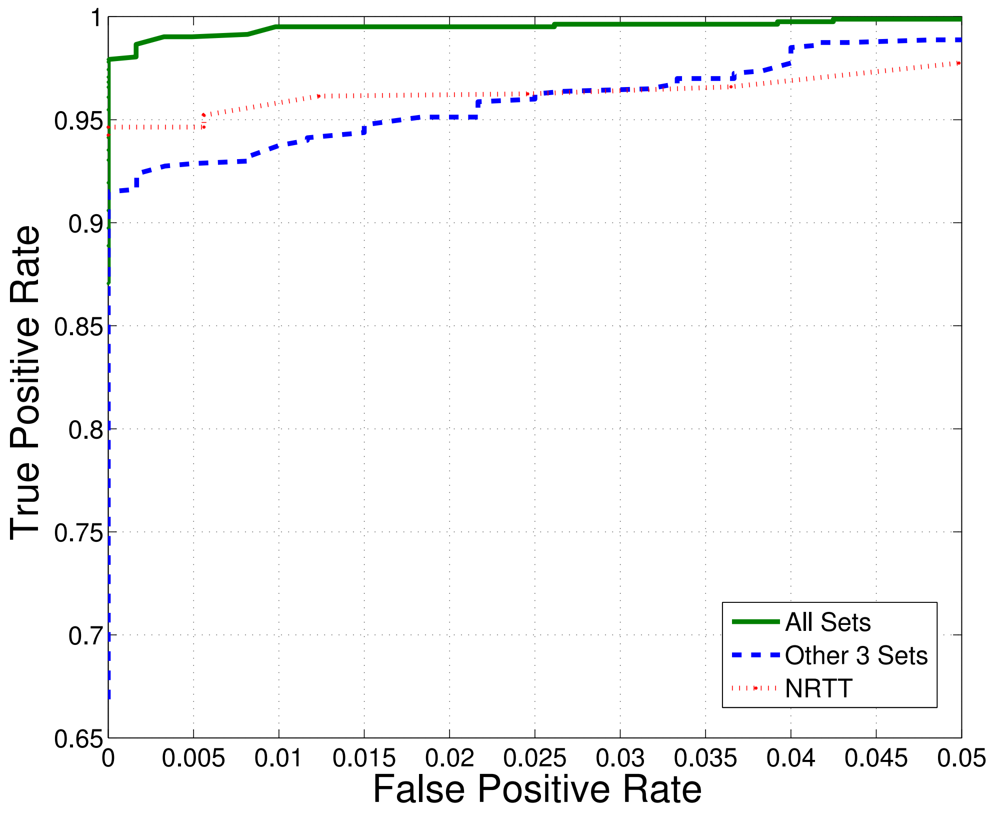


Figure 3.14 ROC curves of FPR-TPR for selected features set, $NRTT$ and the all feature set.

seconds to hundreds of million seconds. It clearly shows that $NRTT$ is able to distinguish between the phishing websites and legitimate website.

Using the same data set, the second test quantitatively shows the detection accuracy in terms of true positive rate (TPR, i.e., the number of correctly detected phishing sites divided by the number of all phishing sites) for different values of error tolerance. As depicted in Figure 3.10, the usage of $NRTT$ for phishing detection can achieve very high true positive rate (i.e., greater than 93% for a 10 ms error tolerance) despite of the detection time periods (The blue curve is measured during night time 00:00 AM and the red curve repeat the same detection process during day time 12:00 PM).

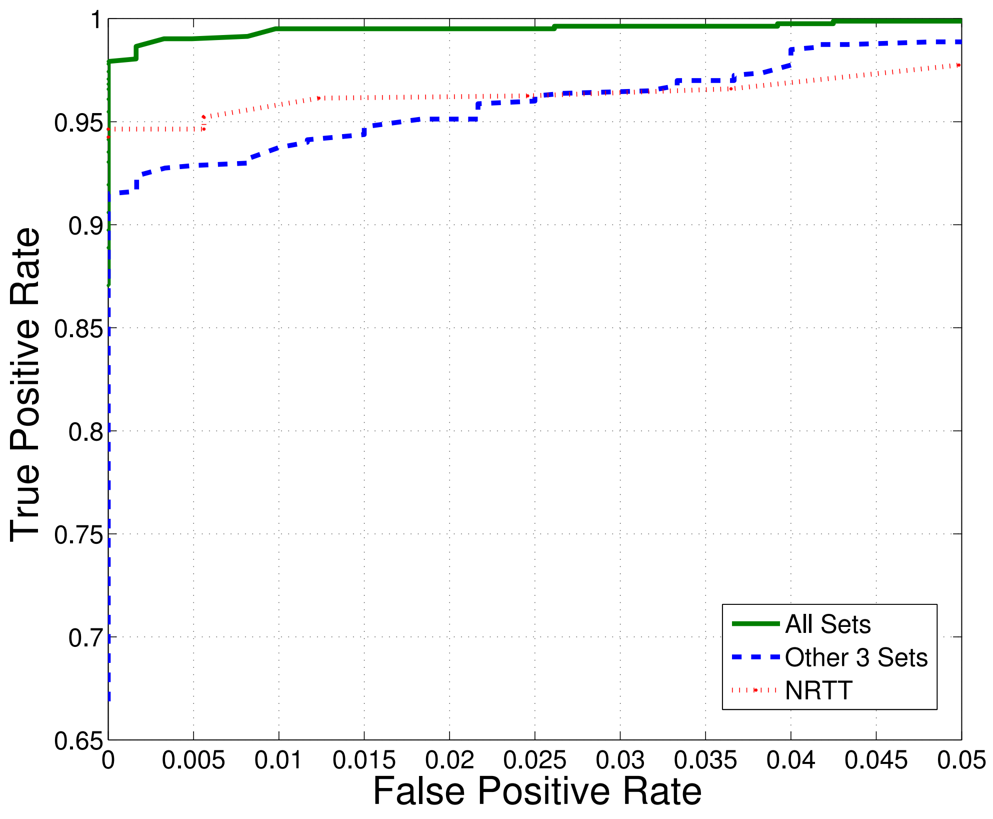


Figure 3.15 True Positive Rate for the website identification algorithm: (1) Text extraction only; (2) Text extraction plus Image Processing.

To alleviate the impact of the network instabilities, we apply the algorithms similar to [61] in processing the raw data of the $NRTT$. These algorithms includes: (1) outliers filtering; and (2) shared increments removal. Figure 3.11 shows 135 real time measured profiles for each of the two profiles of one legitimate website for 60 hours from 9:00 AM Friday to 9:00 PM Sunday. The blue dots indicates the real time profiles without algorithm (1) and (2) while the red small circles are the real time profiles after applying the algorithms. It clearly shows the proposed algorithms successfully alleviate the impact of the network instabilities, thus, it helps to reduce the false positive rate of the system.

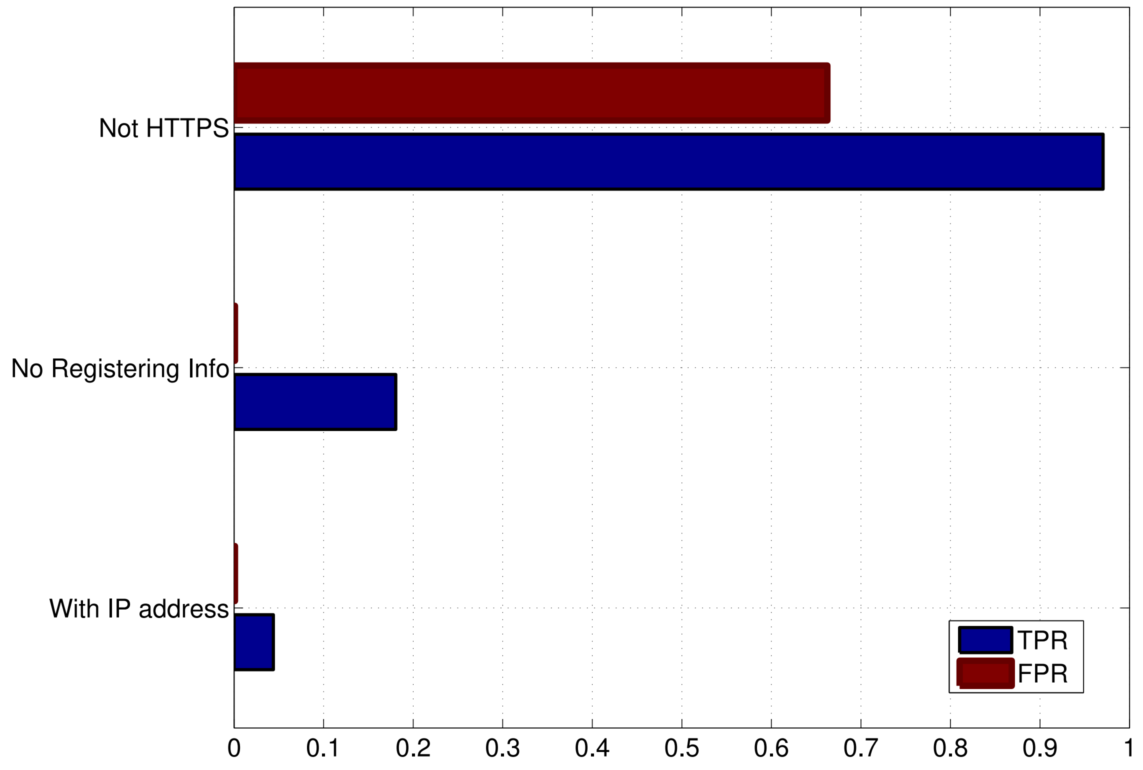


Figure 3.16 True Positive Rate and False Positive Rate for different binary features.

3.5.4 Evaluation of the Target Identification Algorithm

In this test, we use the date set with 851 verified on-line phishing websites (from PhishTank) and 498 legitimate websites (from Alexa) to evaluate the true positive rate (i.e., the number of correctly identified sites divided by the number of all sites) and the false positive rate (i.e., the number of wrongly identified sites divided by the number of all identified websites) of the proposed target identification algorithm.

As depicted in Figure 3.15, with the help of image processing (i.e., extracting text from website images), as detailed in Section 3.4.2), the proposed target identification algorithm can achieve 95% TPR while having a false positive rate less than 5%. We note that, vast majority of the false positives occurred in phishing

Table 3.6 Comparative Evaluation among Different Schemes

Scheme	**Year**	**Data Set**	**TPR**	**FPR**
Guang Xiang et al.	2011	940	93%	0.4%
Colin Whittaker et al.	2010	16,967	90%	0.1%
Kurt Thomas et al.	2011	500,000	87%	0.3%
Gowtham Ramesh et al.	2014	3374	99%	0.3%
Teh-Chung Chen et al.	2014	1945	99%	0.7%
Samuel Marchal et al.	2016	1553	99%	0.1%
Proposed scheme	2016	2000	99%	0.2%

websites (the false positive rate for legitimate website is less than 1%), and hence, it will not affect the TPR of the proposed phishing detection system.

3.5.5 Comparison between the Proposed Framework with State-of-the-art Schemes

Table 3.6 compares the evaluation results of the proposed framework to the most cited state-of-the-art schemes [34] [90] [136] [133] [124] [106]. We note that the a low false positive rate is of great importance for a phishing detection scheme because classifying a legitimate website as a phishing would results in serious consequences [24]. On the other hand, the true positive rate is less important than the false positive rate. In addition, the TPR highly depends on the phishing test set used and we cannot conclude scheme A is better than scheme B only if the TPR of A is 1% higher than the TPR of B. From Table 3.6, we can see that the proposed scheme has comparative performance comparing to the state-of-the-art schemes.

3.6 Conclusion

In this work, we propose a new phishing detection framework by integrating a set of selected phishing detection features. The comparative evaluation of the proposed scheme shows that it has comparable performance results compared to the state-of-the-art schemes. In addition, we introduce a new robust phishing phishing detection feature based on Network Round Trip Time, $NRTT$. The evaluation results show that $NRTT$ can achieve high detection accuracy ($> 93\%$) even if using it alone. To enforce the proposed scheme, we design a novel algorithm to identify the target website from suspicious URL. The experimental results show that the identification accuracy can reach 95%.

CHAPTER 4

ROBUST INSIDER ATTACKS COUNTERMEASURE FOR HADOOP: DESIGN & IMPLEMENTATION

4.1 Introduction

APACHE Hadoop provides an open source framework for the storage and parallel processing of large-scale data sets on clusters of commodity computers. As the amount of data maintained by industrial corporations grows over time, big data processing becomes more important to enterprise data centers. However, the threat of data leaks also continues to grow due to the increasing number of entities involved in running and maintaining cloud infrastructure and operations [140]. The recent boost of big data start-ups such as MongoDB, DataStax, MapR Technologies and Skytree leads to an increased number of points of access in the cloud, that is, larger attack surface for intruders. This can be clearly inferred from a recent report of the Health Information Trust Alliance (HITRUST), which reveals that the total cost of health-care data breach incidents has grown to \$4.1 billion over the recent years [56].

Authentication is the process of gaining assurance that an entity is performing robustly and precisely as intended [132] [33]. In addition, data confidentiality in the cloud is tightly correlated to the user authentication [140]. Therefore, a secure and robust authentication mechanism of both users and services is imperative for secure and private cloud computing and storage operations [86]. However, the continuous growth and the concentration of data in clouds, combined with the increasing adoption of security solutions such as authentication, access control, and encryption drives intruders to be more persistent and creative in developing sophisticated attack strategies [122]. One way to protect clouds and to successfully combat such sophisticated attacks is to push the bar higher through the combination of hardware and software security solutions. Pushing the security down to the hardware level in conjunction with software techniques provides better protection over software-only solutions [104], which is especially feasible and suitable for entity authentication and platform attestation in the cloud.

Currently, Hadoop leverages Kerberos [114] [21] as the primary authentication method and uses DIGEST-MD5 security tokens [80] to supplement the primary Kerberos authentication process, as detailed in Section 4.2. However, in addition to its limitations and security weaknesses, the use of Kerberos for authentication in Hadoop-based environments raises many security concerns. The most vital weakness of Kerberos lies in its dependency on passwords. The session key for data encryption during the initial communication phase with the Key Distribution Center (KDC) is derived from the user's password. Disclosure of KDC passwords allows attackers to capture users' credentials, which turns all Hadoop's security to be useless. The large number of password disclosure incidents through cracking, social engineering, or even database leakage, clearly indicates that this threat is real and pervasive. For example, in 2011, RSA was a target of a spear Phishing attack [118]. A backdoor was installed due to a mistake by an employee who retrieved the Phishing email from her junk mail box and opened it. The malware successfully harvested credentials as confirmed by RSA FraudAction Research Labs. It threatened the security of many important defense contractors like Northrop Grumman, Lockheed Martin, and L-3.

Another important issue of Kerberos lies in its dependency on the KDC which constitutes a single point of failure and even a single point of attack for persistent and dedicated attackers. Although Hadoop's security design introduces delegation tokens to overcome this bottleneck, they lead to a more complex authentication mechanism due to the extra tokens and data flows that are required to enable access to Hadoop services. Many types of token have been introduced, including delegation tokens, block tokens, and job tokens for different subsequent authentications. This, relatively, large number of tokens, not only complicates the configuration and the management of the tokens, but also expands the attack surface [27]. Kerberos keys are stored in an on-line third-party database. If anyone other than the proper user has access to the KDC, through, for example, a malware installation by an insider, the entire Kerberos authentication infrastructure will be compromised and the attacker will be able to impersonate any user [54]. This highlights the fact that insiders could create havoc in Kerberos infrastructure itself, and consequently affect the security posture of the supported Hadoop. It is clear that Kerberos is not well-equipped against insiders or outsiders who could change the execution environment that the user trusts. For example, attackers may install key loggers or other malware-tools to steal users' credentials and data.

In this work, we **design** and **implement** a TPM-based authentication protocol for Hadoop that provides strong mutual authentication services among internal Hadoop entities, in addition to mutually authenticating external clients. Each entity in Hadoop is equipped with a TPM (or vTPM in the case of multi-home virtualized environments) that locks-in the root keys for authenticating the entity to the outside world. In addition to locally hiding the authentication keys and the authentication operations, the TPM captures the current software and hardware configurations of the machine hosting it in an internal set of Platform Configuration Registers (PCRs), as detailed in Section 4.2.3. Using the authentication keys and the PCRs, the TPM-enabled communicating entities are able to establish session keys that can be sealed (decrypted only inside the TPM) and bound to specific trusted platform software and hardware configurations. The bind and seal operations protect against malicious insiders, because they will not be able to change the state of the machine without affecting the values of the PCRs. Our protocol enables remote platform attestation services to clients of third-party, possibly not trusted, Hadoop providers. Moreover, the seal of the session key protects against the ability to disclose the encrypted data in any platform other than the one that matches the trusted configurations specified by the communicating entities. As Figure 4.1 shows, the protocol consists of three overlapping phases: (1) Secure exchange of the session key at the beginning of each communication session; (2) Session key management; and (3) "Fingerprint" attestation mechanism. The details are presented in the subsequent sections.

Our contributions can be summarized as follows:

- Propose a novel TPM-based authentication protocol among Hadoop entities to reduce the risk of the continuously evolving insider attacks, especially due to the proliferation of cloud services and big data applications.
- Propose a periodic remote mechanism for attesting the hardware and software platform configurations. Such attestation mechanism provides hardware level security that supports other software mechanisms in reducing the risk of internal, as well as, external threats.
- Implement the whole system on real world Hadoop platforms and conduct extensive sets of experiments to evaluate the performance overhead and the security merits of our mechanism. The performance and security evaluation clearly shows that our framework provides better security guarantees with acceptable performance overhead compared to the state-of-the-art Kerberos-based implementations.
- Provide a thorough theoretical analysis using the BAN logic to rigorously prove the correctness and the trustworthiness properties of our authentication protocol.

4.2 Background

In this section, we introduce necessary information about Hadoop architecture, state-of-the-art Hadoop security design and the basis of Trusted Platform Module technology (TPM).

4.2.1 Hadoop Structure

As depicted in Figure 4.2, Hadoop clusters have three major categories of server roles: (1) Client machines, (2) Master nodes, and (3) Slave nodes. The role of the client

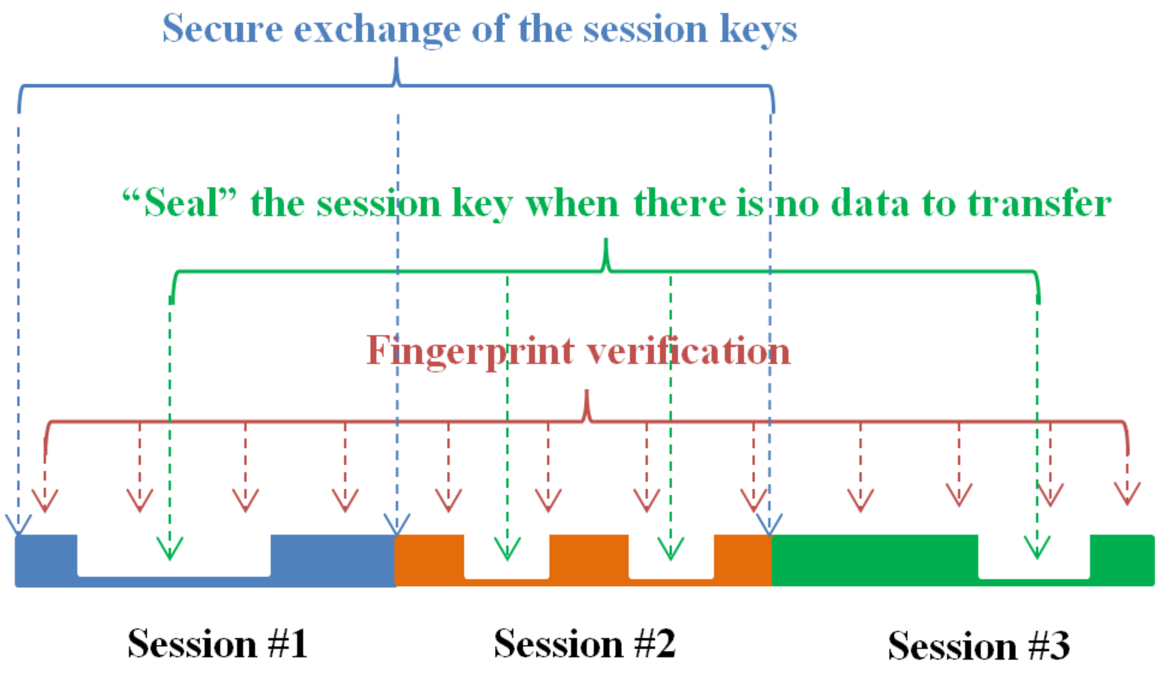

Secure exchange of the session keys
"Seal" the session key when there is no data to transfer
Fingerprint verification
Session #1
Session #2
Session #3

Figure 4.1 Overlapping phases of the TPM-based authentication protocol.

machine is to load the data into the cluster, to submit MapReduce jobs describing how data should be processed, and to fetch or view the results of the task when processing finishes. The Master nodes (i.e., NameNode, Secondary NameNode and JobTracker) supervise the two major components of Hadoop, namely, the distributed data storage (HDFS), and the distributed data processing (MapReduce) [30] [29]. The NameNode is responsible for coordinating data storage operations when a client machine requests to load the data into HDFS, while the JobTracker is in charge of supervising parallel MapReduce processing. The slave nodes (i.e., DataNodes and TaskTrackers) are responsible for storing the data and executing the computational tasks assigned by the Master nodes, respectively.

4.2.2 Hadoop Security Design

Hadoop uses Kerberos for its authentication operations [80]. The complete authentication process is illustrated in Figure 4.3. The client obtains a Delegation Token (DT) through initial Kerberos authentication (step 1). When the client uses the DT

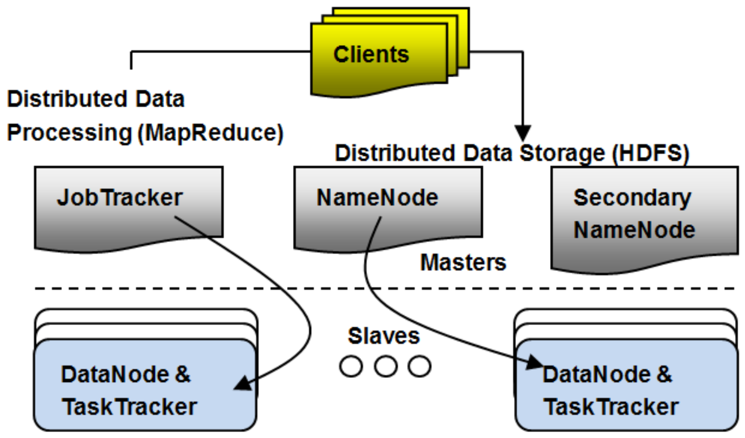


Figure 4.2 Hadoop architecture based on server roles.

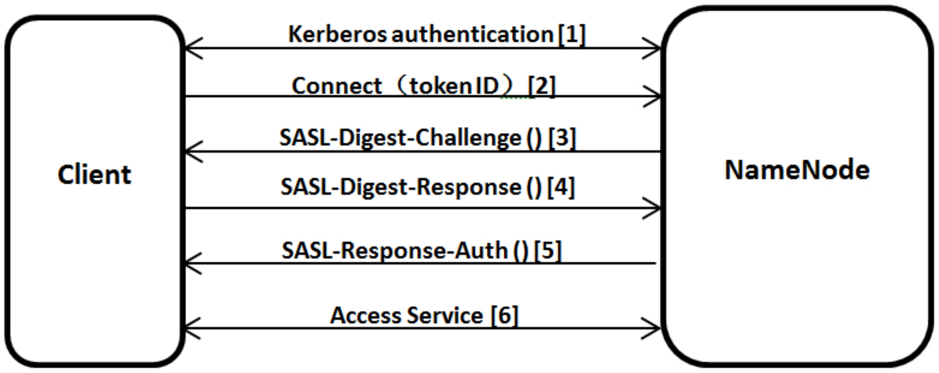

Client
NameNode
Kerberos authentication [1]
Connect（tokenID）[2]
SASL-Digest-Challenge () [3]
SASL-Digest-Response () [4]
SASL-Response-Auth () [5]
Access Service [6]

Figure 4.3 Authentication process in Kerberos-based Hadoop.

to authenticate, she first sends the ID of the DT to the NameNode (step 2). Then, the NameNode checks if the DT is valid. If the DT is valid, the client and NameNode try to mutually authenticate using their own Token Authenticators, which are contained in the delegation tokens, as the secret key using DIGEST-MD5 protocol (steps 3-6) [130].

4.2.3 Trusted Platform Module (TPM)

TPM is a secure crypto-processor, which is designed to protect hardware platforms by integrating cryptographic keys into devices [107]. It has been designed with the goal to enhance platform security through mechanisms that are beyond the capabilities of today's software-based systems [13].

TPM supports three main services: (1) The remote attestation service creates a nearly un-forgeable hash-key summary of the hardware and software configurations in a way that allows other parties to verify the integrity of the software and the hardware. (2) The binding service encrypts data using the TPM endorsement key, a unique RSA key burned into the chip during its production, or another trusted key descended from it. (3) The sealing service encrypts data in a similar manner to binding, but it additionally specifies the state that the TPM must be in for the data to be unsealed (decrypted) [107].

The platform configuration register (PCR) of the TPM is a 160-bit storage location that is used to record aggregate software and hardware measurements of the platform, which include: (1) BIOS, ROM, Memory Block Register [PCR index 0 - 4]; (2) Operating System (OS) loaders [PCR index 5 - 7]; (3) Operating System (OS) [PCR index 8 - 15]; (4) Debug [PCR index 16]; (5) Localities, Trusted OS [PCR index 17 - 22]; and (6) Applications specific measurements [PCR index 23] [4].

The TPM is capable of creating an unlimited number of Attestation Identity Keys (AIK). Each AIK is an asymmetric key pair that is used for signing data that is internally generated by the TPM, such as the storage key. A storage key is derived from the Storage Root Key (SRK), which is embedded in the TPM chip during the manufacturing process. Using the generated storage key along with the PCR values, one could perform sealing to bind the data to a certain platform state (i.e., a specific platform software and hardware configuration). The encrypted data could only be unsealed (decrypted) under the same PCR values (i.e., the same platform state).

To date, more than 500 million PCs have been shipped with TPMs, an embedded crypto capability that supports user, application, and machine authentication with a single solution [104]. Additionally, many virtual TPM implementations exist for virtualized environments [32] [110].

4.2.4 Integrity Measurement Architecture

Co-operating with the hardware TPM, the integrity measurement architecture (IMA) (proposed by IBM [17]) provides an efficient measurement system for dynamic executable content. IMA provides real time measurements of the platform (user applications, OS libraries, etc.) during the post-boot period, while the TPM enables the pre-boot measurements. In this dissertation, we assume the IMA is pre-configured and installed on each platform.

4.3 TPM-based Hadoop Authentication Framework

In this section, we first introduce the attack model, then, we present the design of the proposed TPM-based Hadoop authentication protocol.

The protocol uses the key binding of the TPM to secure the exchange and management of the session keys between any two Hadoop communicating entities (NameNode/JobTracker, DataNode/TaskTracker and Client). To achieve this, we assume that every Hadoop entity has a TPM. Figure 4.4 (a) depicts the high-level processes of the protocol which are detailed in the following subsections. The protocol consists of two processes, the certification process and the authentication process.

Note that the public key infrastructure (PKI) is only used to secure the exchange of the symmetric session keys. The expensive certification and management process is only used during the certification process, where the cost amortized through the use of TPM AIK functionality as explained in Section 4.3.2.

4.3.1 Attack Model

In addition to the traditional external threats, we believe that clouds are more susceptible to internal security threats, especially from other untrusted users [92]

[123]. Many enterprises are likely to deploy their data and computations across different cloud providers for many reasons including load balancing, high availability, fault tolerance, and security, in addition to avoiding single-point of failure and vendor lock-in [63] [44] [108]. However, such behavior increases the attack surface and the probability of compromise of Hadoop entities. For example, a DataNode may get infected with a malware that makes it unsafe to operate on sensitive data. Therefore, it is imperative for any security solution for Hadoop to enable the detection of any unauthorized change in the software and hardware configurations of its entities. Our entity authentication protocol is specifically designed to counter such adversarial actions, assuming an attack model with the following attacker capabilities and possible attack scenarios:

- **Attacks against TPM**: We do not address the attacks against TPM (e.g., side channel timing attack [117]).Thus, we assume attackers can not compromise the TPM chip or its functions.
- **Attacks against slave nodes or client machines**: We assume that attackers are capable of installing (directly or remotely) malware or making malicious hardware changes in the compromised slave nodes (e.g., the DataNodes) or client machines.
- **Attacks against master nodes**: We assume that attackers are capable of installing (physically or remotely) malware or making malicious hardware changes in the master nodes (e.g., the NameNode). However, this capability could be revoked if the NameNode is deployed in a trustworthy environment (e.g., a private cloud with strict security policy enforced), as detailed in Section 4.3.4.
- **Denial of Service attacks (DoS attacks)**: We do not address DoS attacks.

4.3.2 TPM-Based Hadoop Certification Process

The certification process (similar to that presented in [7]) is triggered by the client or the NameNode and is depicted in Figure 4.4 (b). The client in this work refers to any entity that interacts with the NameNode such as a user submitting a job or a DataNode. The TPM of the client/NameNode creates a RSA key using the SRK as a parent. This key is used as the AIK. The AIK is then certified by a PCA. This process is a onetime pre-configuration operation that takes place once during the initialization of the TPM. The client's/NameNode's TPM then creates a binding key that is bound to a certain platform. Then the TPM seals the private part of the binding key with a certain PCR configuration. Finally, the client/NameNode uses the AIK to certify the public part of the binding key. Once the AIK is certified by the PCA, it can be used to sign all types of keys generated by the TPM without referring back to the PCA, which greatly reduces the communication overhead.

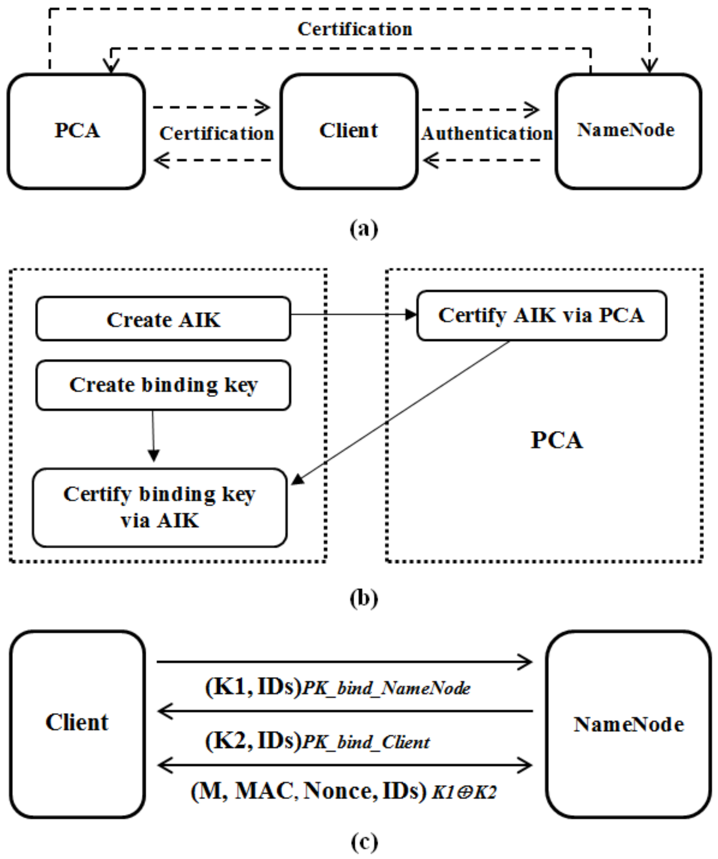


Figure 4.4 (a)The high level processes of our TPM-based Hadoop authentication protocol (Client to NameNode in this example); (b)TPM-based Hadoop certification process; and (c)TPM-based Hadoop authentication process.

4.3.3 The Authentication Process

The authentication process (cf. Figure 4.4 (c)) implements the mutual authentication between the NameNode and the client. The Client sends a random number K_1 along with the corresponding IDs (e.g., $remoteID$ in Hadoop codes) to the NameNode. This message is encrypted by the public binding key of the NameNode. The NameNode sends a random number K_2 along with the corresponding ID to the client. This message is encrypted by the public binding key of the client. Using K_1 and K_2, both the client and the NameNode generate the session key $key\_session = K_1 \oplus K_2$. Note that only the correct NameNode can obtain K_1 by decrypting the message sent by the client using the NameNode's SK_bind, which is bound to the target NameNode's TPM with a certain software and hardware configuration (sealed binding key). Similarly, only the correct client can obtain K_2 by decrypting the message sent by the NameNode using the client's SK_bind, which is bound to the client's TPM with the corresponding software and hardware configurations. This ensures mutual authentication between the client and the NameNode.

The newly exchanged session key is then locked into a certain PCR value in an operation known as seal operation using the TPM's command "Seal". Seal takes two inputs, the PCR value and the session key ($Seal(PCRindexes, Key\_session)$). This ensures that $key\_session$ can only be decrypted using the hardware secured keys of the TPM in that particular platform state. By sealing the session key to specific acceptable hardware and software configurations (i.e., specific PCRs value), we protect against potential tampering with the firmware, hardware, or software on the target machine (e.g., through malware installations or added hardware/software key loggers). Moreover, the session key ($key\_session$) is made to be valid only for a predefined period of time, after which the session key expires and the authentication process has to be restarted to establish a new session key as needed. The validity period of the session key is an important security parameter in our protocol. Short validity periods provide better security in the case of session key disclosure since fewer communications are exposed by disclosing the key. However, shorter periods incur extra overhead in establishing more session keys.

Additionally, a nonce is added to every message to prevent replay attacks. Finally, message authentication codes (MAC) are included with each message to ensure data integrity. The communication message format is as follows:

$$(Message, MAC, Nonce = Nonce + +, IDs)key\_session$$

.

4.3.4 Periodic Fingerprint Checking (Cross-Platform Authentication)

In a non-virtualized environment, the TPM specification assumes a one-to-one relationship between the OS and the TPM. On the other hand, virtualized scenarios assume one-to-one relationship between a virtual platform (virtual machine) and a virtual TPM [129]. However, Hadoop systems employ a master/slaves architecture. The NameNode is the master that manages many DataNodes as slaves. If the number of DataNodes grows, the number of session establishment processes that the NameNode is involved in also grows. Each session involves many TPM operations (e.g., seal and unseal). For large systems, the TPM may become a bottleneck due to the limitation of one TPM/vTPM per NameNode according to current implementations of TPM/vTPM.

To address this practical issue and alleviate the potential performance penalty of TPM operations, we introduce the concept of periodic platform-Fingerprint checking mechanism based on the heartbeat protocol in Hadoop (cf. Figure 4.5). The idea is to offload most of the work from the TPM of the NameNode to the NameNode itself. However, this requires us to loosen our security guarantees and change the attack model by assuming that the NameNode is "partially" trusted. We argue that this assumption is reasonable for the following reasons: (i) Hadoop deployment usually involves one or few NameNodes [119], and hence, it is plausible and affordable to secure them in a trusted environment. For example, an enterprise can secure its NameNode by deploying it in the enterprise's local data center or in a high-reputation cloud platform with strict security standards. While the DataNodes can be deployed in environments with less strict security requirements. (ii) Our protocol is designed to limit the potential security damage of untrusted NameNode. A NameNode that gets compromised (that is, its software and/or hardware configuration is changed illegally) will only stay unnoticed for a short time, because other parties (such as DataNodes and clients) are designed to randomly request attestation of the authenticity of the NameNode. In on-demand attestation, an interacting entity with the NameNode asks the NameNode to send a TPM-sealed value of its current software and hardware configurations. If the requesting entity receives the right values for the PCR of the NameNode within a predefined time, then the NameNode is trusted; otherwise, a suspicious alert is raised about the trustworthiness of the NameNode.

The platform Fingerprints (i.e., PCR values) of each Hadoop entity that interacts with the NameNode (e.g., DataNode) are collected a priori and stored in the NameNode. This can be achieved during the registration process of the entity with the NameNode. The heartbeat protocol in Hadoop periodically sends alive information from one entity to another (e.g., from DataNode to NameNode). We leverage this native Hadoop feature by configuring each Hadoop entity to periodically (or on-demand) send the new PCR values (modified by PCR extension operations) to the NameNode for consistency checking with the stored PCR values. The TPM in the interacting entity signs its current PCR values using its AIK key and sends the message to the NameNode. When the NameNode receives the signed PCR values, it verifies the signature, and if valid, it compares the received values with the trusted pre-stored values. If a match is found, the authentication succeeds and the session continues. Otherwise, the authentication fails and penalty may apply (e.g., clear up the session key, shut down the corresponding DataNode, etc.). By doing so, the number of NameNode TPM operations decreases significantly as we replace the TPM seal and unseal operations with the Fingerprint verification that is performed outside the TPM (cf. Figure4.5).

However, there is a tradeoff between the security guarantees and the interval of the Fingerprint verification process, which reflects the extra overhead of the system. In other words, the interval value depends on the user's security requirements and is application dependent.

So far we have assumed that the NameNode is partially trusted and argue in favor of that. Nevertheless, in systems that require higher security guarantees and that can afford redundancy, we can eliminate the assumption of partial trusted NameNode. Untrusted NameNode can be neutralized by borrowing concepts from the fault tolerance domain. Multiple redundant NameNodes can be deployed and the NameNode attestation can be achieved through majority voting among the responses of the different NameNodes [111, 66, 76]. In fact, multiple NameNodes have been used in Hadoop implementations to scale up services such as directory, file and block management [119]. We can leverage such deployment (or deploy new NameNodes if necessary) to implement majority voting on the process of periodic platform-Fingerprint checking mechanism. Correct NameNode operations can be achieved as long as the total number of NameNodes is more than $2n$, where n is the number of compromised NameNodes (assuming no faulty nodes). For example, given 5 NameNodes that are independently running on different clouds, the attestation decision can be made based on the majority voting among them. Under this scenario, even if two of the NameNodes are compromised, the attestation decision will still be the correct one.

4.3.5 Security Features

The security features of our design include: (1) *Session key binding.* The session key is generated by $XORing$ a local and an external random numbers (K_1 and K_2). This ensures that only the party that has the appropriate private portion of the binding key will be able to decrypt the message and get the external random number. Furthermore, the decryption keys exist only inside the TPM chip and are sealed to specific hardware and software configurations. This would protect against potential malicious insiders as they will not be able to know anything about the session key. (2) *Session key sealing.* The session key is sealed with TPM functions. The sealed session key can be decrypted only under the same platform conditions (as specified by the PCR values) using the associated sealing key that resides inside the TPM. (3) *Periodic Fingerprint attestation mechanism.* This guards against malicious users who attempt to change the execution environment in a DataNode (by, for example, installing malware/spyware) in a bid to compromise data confidentiality.

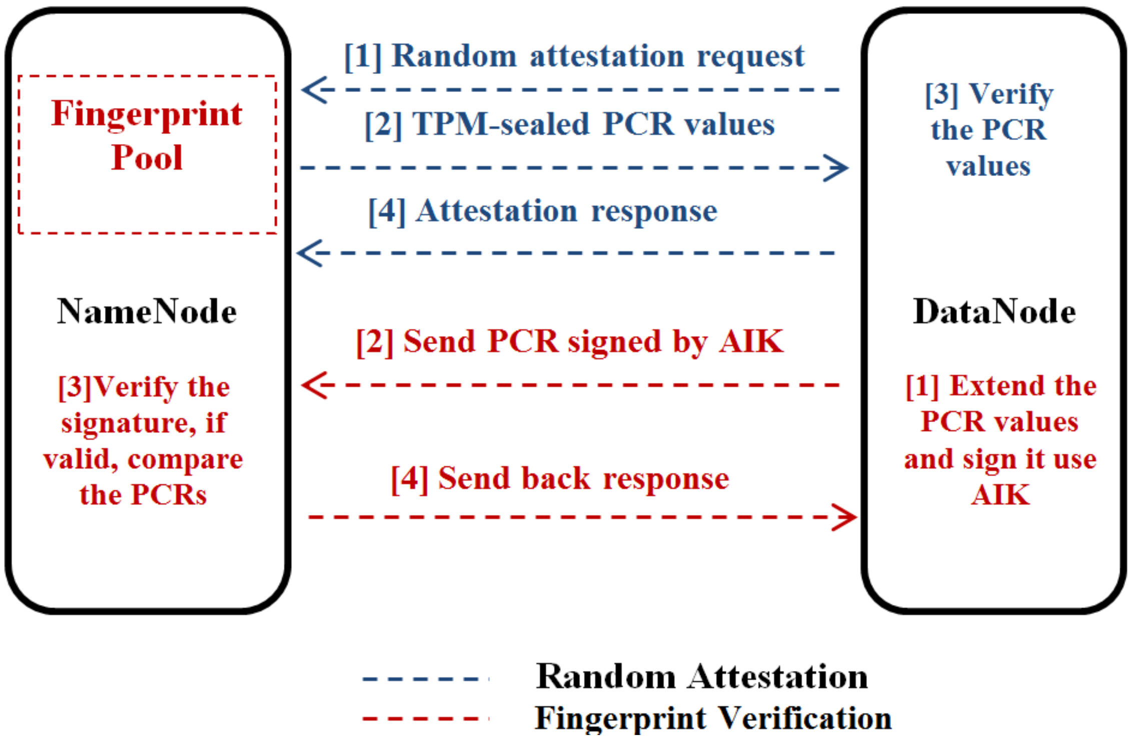
4.4 Formal Security Analysis

We use Burrows-Abadi-Needham (BAN) logic ([7]) to formally prove the following two properties of the proposed authentication protocol:

- **Correctness**: Implies that the protocol performs as intended, that is, two legitimate parties should always correctly authenticate.
- **Trustworthiness**: Implies that the protocol is secure under the attack model defined earlier, that is, only legitimate parties can successfully authenticate.

  The proof sketch is shown as follows:

1. Present the notations and postulates/rules of BAN logical.
2. Generate a simplified version of the original protocol.
3. Idealize the simplified protocol.



Fingerprint Pool
NameNode
[3]Verify the signature, if valid, compare the PCRs
[1] Random attestation request
[2] TPM-sealed PCR values
[4] Attestation response
[2] Send PCR signed by AIK
[4] Send back response
[3] Verify the PCR values
DataNode
[1] Extend the PCR values and sign it use AIK
Random Attestation
Fingerprint Verification

Figure 4.5 Illustration of the random attestation and the periodic Fingerprint verification.

4. Symbolize the assumptions.
5. Formalize the goals.
6. Present the formal prove of the goals.

Recall that the proposed protocol (cf. Figure 4) consists of two main phases: (1) the certification phase and (2) authentication phases. The simplified objective of the protocol is to correctly and securely exchange the necessary keys among legitimate entities in the presence of the defined attackers. Therefore, we first prove that the public keys distributed by the PCA correspond to the claimed owners, that is, the mutual authentication parties believe in the public keys of each other, and that the random numbers (K_1 and K_2) are securely exchanged. Second, we need to prove that only legitimate entities can compute the session keys, that is, an entity can get the session key if and only if it maintains the same hardware and software state bound to that key.

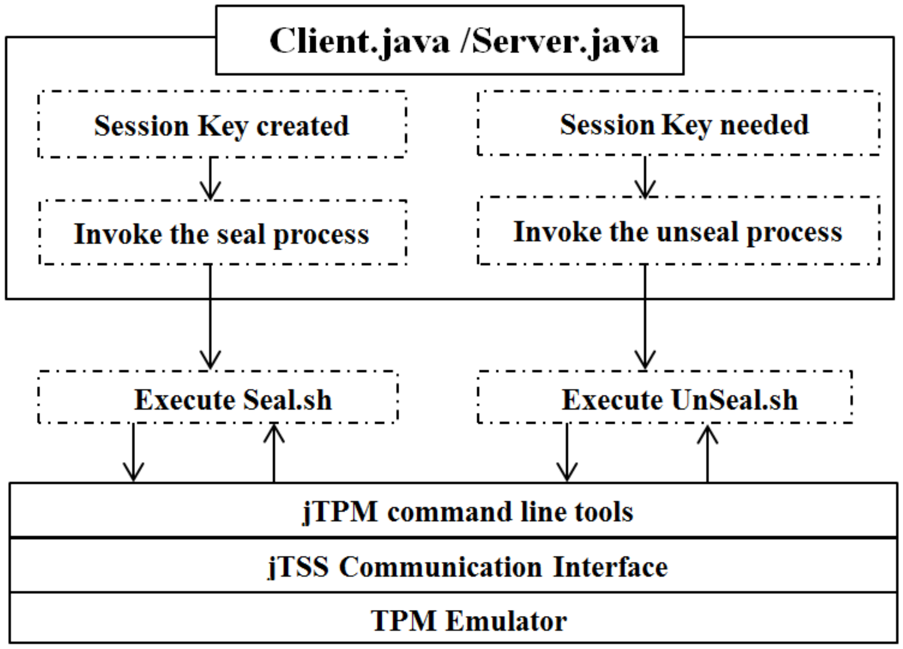


Figure 4.6 Flow of seal and unseal operations within the various TPM-based Hadoop implementation layers.

4.4.1 Notations and Postulates/Rules of BAN Logical

Following is a list of the BAN logical notations that we use in our proof steps:

Basic notations:

- C, N, P denote the client, the NameNode and the PCA.
- K_1 and K_2 are random numbers.
- K_{cn} is the computed symmetric session key, which is only known by C and N, where, $K_{cn} = K_1 \oplus K_2$.
- K_c, K_n, K_p are the public keys of C, N, and P.
- $K_c^{-1}, K_n^{-1}, K_p^{-1}$ are the private keys of C, N, and P.
- X or Y : represents a formula (e.g., a message sent).

In addition to the traditional basic notations, we introduce two new basic notations that represent the hardware and software configurations (recorded in the PCR values of the TPM):

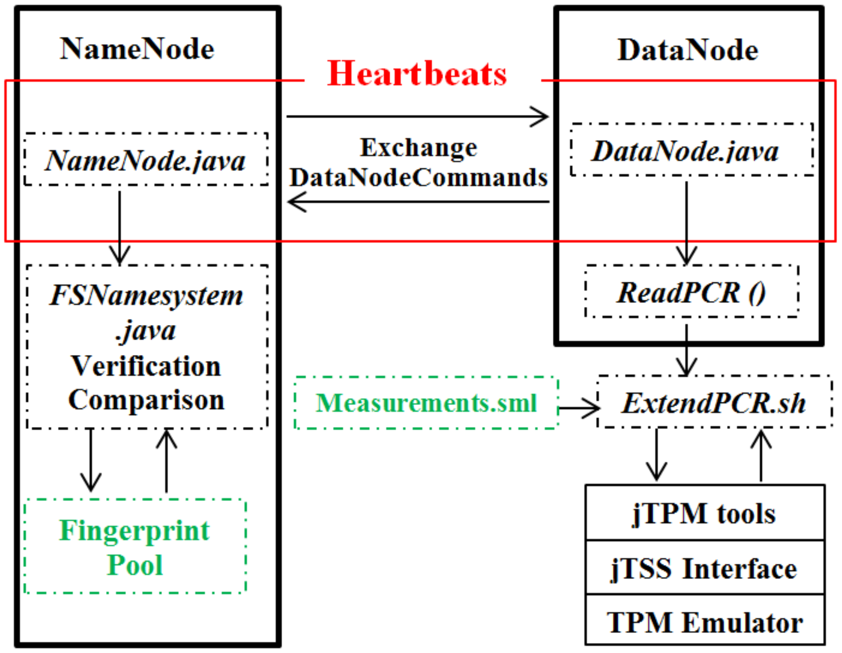


Figure 4.7 Illustration of the detailed process of periodic Fingerprint checking mechanism.

- U and U' represent the TPM combined hardware and software initial state and the state when authenticating, respectively.

Logical notations:

- $C \mid\equiv X$: C believes/would be entitled to believe X;
- $C \triangleleft X$: C sees X. Someone has sent a message containing X to C, who can read and repeat X;
- $C \mid\sim X$: C said X. The principal C at some time sent a message including the statement X;
- $C \mid\Longrightarrow X$: C controls X. The principal C is an authority on X and should be trusted on this matter;
- $C \overset{X}{\rightleftharpoons} N$: X is only known by C and N;
- $C \overset{K_{cn}}{\leftrightarrow} N$: K_{cn} is the key shared by C and N;
- $\#(X)$: X is fresh. X has not been sent in a message at any time before the current run of the protocol;
- $\{X\}_{K_{cn}}$: X is encrypted by K_{cn};

- $\overset{K_c}{\mapsto} C$: K_c is a public key of C;
- $\frac{X}{Y}$: if X is true, then Y is true.

Ban logical postulates: BAN logic consists of many logical rules. These pre-agreed rules are used as theorems in the deduction. For the purpose of the current proof, we use 4 of the existing rules, in addition to a new TPM unsealing rule. Note that every C in the rules could be replaced by N.

– The *Message meaning rule* for the interpretation of messages. Here, we only use the rule for public keys.

$$\frac{C \mid\equiv \overset{K}{\mapsto} P, P \triangleleft \{X\}_{K^{-1}}}{C \mid\equiv P \mid\sim X}$$

That is, if C believes that K is P's public key, and C receives a message encoded with P's secret key, then C believes P once said X.

– The *nonce-verification* rule expresses the check that a message is recent, and hence, that the sender still believes in it:

$$\frac{C \mid\equiv \#(X),\ C \mid\equiv P \mid\sim X}{C \mid\equiv P \mid\equiv X}$$

That is, if C believes that X could have been uttered only recently and that P once said X, then C believes that P believes X.

– The *jurisdiction* rule states that if C believes that P has jurisdiction over X, then C trusts P on the truth of X:

$$\frac{C \mid\equiv P \mid\Longrightarrow X,\ C \mid\equiv P \mid\equiv X}{P \mid\equiv X}$$

– If a principal sees a formula, then he also sees its components, provided that he knows the necessary keys:

$$\frac{C \mid\equiv \overset{K^{-1}}{\mapsto} N,\ C \triangleleft \{X\}_K}{P \triangleleft X}$$

– If a principle's hardware and software state equals to the initial state, then he is able to unseal the content he sealed before.

$$\frac{C(U') == C(U), C \triangleleft \{X\}_{seal}}{C \triangleleft X}$$

4.4.2 The Simplified Version of the Original Protocol

The original authentication protocol (ref. fig 5) can be simplified as follows:

Step 1: C and N get each other's public key from P:

$$P \rightarrow C : \{K_n, N\}_{K_p^{-1}}; \quad P \rightarrow N : \{K_c, C\}_{K_p^{-1}}$$

Step 2: C and N exchange two random numbers K_1 and K_2 to generate the session key.

$$C \rightarrow N : \{K_1, C\}_{K_n}; \quad N \rightarrow C : \{K_2, N\}_{K_c}$$

Step 3: C and N unseal K_c^{-1} and K_n^{-1} to extract K_1 and K_2 if the software and hardware configuration do not change, then the shared session key $K_cn = K1 \oplus K2$ can be generated and used for later communication.

$$C \text{ gets } K_2; \quad N \text{ gets } K_1$$

4.4.3 The Idealized Protocol

Now we idealize the original protocol to the standard form of BAN logic:

Step 1: $P \rightarrow C : \{\overset{K_n}{\mapsto} N\}_{K_p^{-1}};$

Step 2: $P \rightarrow N : \{\overset{K_c}{\mapsto} C\}_{K_p^{-1}};$

Step 3: $C \rightarrow N : \{K_1\}_{K_n};$

Step 4: $N \rightarrow C : \{K_2\}_{K_c};$

Step 5: $C \triangleleft K_2, N \triangleleft K_1$.

4.4.4 Symbolizing the Assumptions

We formalize the assumptions of the proposed protocol as follows:

$$C \mid\equiv \overset{K}{\mapsto} C,\ N \mid\equiv \overset{K}{\mapsto} N,\ C \mid\equiv \overset{K_p}{\mapsto} P,\ N \mid\equiv \overset{K_p}{\mapsto} P$$

$$P \mid\equiv \overset{K_c}{\mapsto} C,\ P \mid\equiv \overset{K_n}{\mapsto} N,\ P \mid\equiv \overset{K}{\mapsto} P$$

$$C \mid\equiv (P \mid\Longrightarrow \overset{K_n}{\mapsto} N),\ N \mid\equiv (P \mid\Longrightarrow \overset{K_c}{\mapsto} C)$$

$$C \mid\equiv \#(K_1),\ N \mid\equiv \#(K_2)$$

$$C \mid\equiv C \overset{K_1}{\rightleftharpoons} N,\ C \mid\equiv C \overset{K_2}{\rightleftharpoons} N$$

$$C \mid\equiv \#(\overset{K_n}{\mapsto} N),\ N \mid\equiv \#(\overset{K_n}{\mapsto} N)$$

C and *N* know the public key of P, as well as their own keys. In addition, *P* knows his own keys, and the public keys of *C* and *N*. *C* and *N* trust *P* to correctly sign certificates giving the public keys of them. Also, *C* and *N* believe the random numbers that they generate are fresh, secure. Last but not least, *C* and *N* assume that the message containing the public key of each other is fresh.

4.4.5 Formalization of the Security Goals

The security goals of the proposed protocol can be formalized as:

Goal 1: *C* and *N* trust each other's public keys.

$$C \mid\equiv \overset{K_n}{\mapsto} N,\ N \mid\equiv \overset{K_c}{\mapsto} C$$

Only if *C* and *N* believe in the public keys of each other, then the two random numbers K_1 and K_2 can be securely exchanged.

Goal 2: *C* and *N* both get the session key if they maintain the same hardware and software state.

$$\frac{C(U') == C(U) \text{ and } N(U') == N(U)}{C \overset{K_{cn}}{\rightleftharpoons} N}$$

4.4.6 Formal Proof

Proof of Goal 1: $\because$ the assumption: $C \mid\equiv \overset{K_p}{\mapsto} P$, and the protocol step 1: $C \triangleleft \{\overset{K_n}{\mapsto} N\}_{K_p^{-1}}$, according to rule (1):

$$\therefore \quad C \mid\equiv P \mid\sim \overset{K_n}{\mapsto} N \tag{1}$$

$\because$ the assumption: $C \mid\equiv \#(\overset{K_n}{\mapsto} N)$, and equation (1), according to rule (2):

$$\therefore \quad C \mid\equiv P \mid\equiv \overset{K_n}{\mapsto} N \tag{2}$$

$\because$ the assumption: $C \mid\equiv (P \mid\Longrightarrow \overset{K_n}{\mapsto} N)$, and equation (2), according to rule (3):

$$\therefore \qquad C \mid\equiv \overset{K_n}{\mapsto} N$$

Similarly, we can prove that: $N \mid\equiv \overset{K_c}{\mapsto} C$ Therefore, **Goal 1** is verified.

Proof of Goal 2: $\because$ C sealed its private key K_c^{-1} himself (i.e., $C \triangleleft \{K_c^{-1}\}_{seal}$), according to rule (5):

$$\therefore \qquad \frac{C(U') == C(U)}{C \triangleleft K_c^{-1}} \tag{3}$$

$\because$ The assumption: $C \mid\equiv \overset{K}{\mapsto} C$, the protocol step 4: $C \triangleleft \{K_2\}_{K_c}$ and equation(3), according to rule (5):

$$\therefore \qquad \frac{C(U') == C(U)}{C \triangleleft K_2} \tag{4}$$

$\because$ C generates K_1 and equation (4).

$$\therefore \qquad \frac{C(U') == C(U)}{C \triangleleft (K_{cn} = K_1 \oplus K_2)} \tag{5}$$

Similarly, we can prove that:

$$\frac{N(U') == N(U)}{N \triangleleft (K_{cn} = K_1 \oplus K_2)} \tag{6}$$

Consequently, based on equation (5) and equation (6), we can conclude that:

$$\frac{C(U') == C(U) \text{ and } N(U') == N(U)}{C \overset{K_{cn}}{\rightleftharpoons} N}$$

Therefore, **Goal 2** is verified.

4.5 Implementation

4.5.1 Implementation of the Authentication Protocol

To efficiently manage the complex communication flows, Hadoop utilizes RPC-Dynamic Proxy that creates a simple interface between one client and one server [6]. We divide the implementation of our authentication protocol into three sub-tasks.

Task 1: Includes the exchange of two random numbers K_1 and K_2 between the client and the server. In general, multiple "calls" (i.e., different HDFS/MapReduce commands or operations) may occur within one RPC connection, and we use the first call in each RPC connection to exchange K_1 and K_2. For sending K_1 from the client to the server, we create a new variable *Call.connection.*K_1 in the RPC connection header field, and then use $WriteRPCHeader()$ function in $Client.Java$ to forward it to the server. The server then reads K_1 in *ReadAndProcess*() function. For sending K_2 from the server to client, we also create a new variable *Call.connection.*K_2 and conditionally (i.e., if K_2 has never been sent) forward it to the client via *SetupResponse*() function in $Server.java$. The client then decodes K_2 in *ReceiveResponse*() function.

Note that both K_1 and K_2 are encrypted using the corresponding receiver's public binding key and decrypted via their sealed private binding key which is bound to a certain platform configuration.

Task 2: Includes the dynamic protection of the *Session_Key* $= K_1 \oplus K_2$ using TPM seal and unseal operations. After securely exchanging the two random numbers K_1 and K_2, each of the client and the server generates its own copy of the *Session_Key*. Right after the generation of the *Session_Key*, a java runtime process is invoked to execute a shell script (i.e., *seal.sh*) that immediately seals the *Session_Key* using jTPM commands via jTSS java interface. Whenever there is a need to use the *Session_Key*, the client/server invokes another java runtime process to conduct a shell script (i.e., *unseal.sh*) to unseal the *Session_Key* for encryption or decryption of the data. Figure 4.6 illustrates the flow of seal and unseal operations within the various TPM-based Hadoop implementation layers.

Task 3: Includes the management and synchronization of the security credentials (e.g., *Session_Key* , *Sealed_Session_Key* and *Unsealed_Session_Key*). In order to efficiently and securely manage the communication credentials, we build a management and synchronization mechanism for the TPM-based Hadoop. Firstly, we distinguish the *Session_Key* by the users (e.g., hdfs, mapred) in the same platform. Since Hadoop utilizes RPC dynamic proxy to simplify the communication, the RPC connections of the same user could share the same *Session_Key*. This mechanism greatly reduces the number of seal/unseal operations while maintaining the same security level. Secondly, since many RPC connections share the same *Session_Key*, synchronization issues arise (i.e., simultaneous accesses to the same *Session_Key*). To handle such issues, we create a file called *session_key_list* that records the IDs of all the *Session_Keys* that currently exist. The client/server checks the list before creating or using the *Session_Key*, and locks the corresponding *Session_Key* while using it. Thirdly, we define different access control policy and lifetime for different security credentials. The *Session_Key* has the shortest lifetime (i.e., deleted right after sealing it) and could only be accessed by the user who created it and the TPM owner who seals it. The *Sealed_Session_Key* holds the longest lifetime (its lifetime could be adjusted according to the user's security requirements) and could only be accessed by the owner of the TPM (i.e., the one who knows the password of the TPM). The *Unsealed_Session_Key* keeps the medium lifetime (depends on the user's

Table 4.1 Access Control Policy and Lifetime for Different Security Credentials

	Session Key	**Sealed Session Key**	**Unsealed Session Key**	**TPM Sealing Key**	**Session Key List**
Lifetime	Shortest	Longest	Medium	Permanent	Permanent
Access Control	All Users	TPM Owner	User	TPM Owner	All Users

security requirements) and could only be accessed by the user of the corresponding *Session_Key*. Furthermore, the sealing key (i.e., used for seal/unseal operations) is well protected by the TPM and can only be accessed by the owner of TPM. In addition, the *session_key_list* only contains the IDs of the *Session_Keys*, thus knowing the contents of the *session_key_list* will not help the attacker to obtain the *Session_Key*. Table 4.1 shows the access control and lifetime for different security credentials.

4.5.2 Heartbeat Protocol Implementation

As mentioned in Section 4.3.4, in order to offload the work of the TPM on the NameNode, we introduce the periodic Fingerprint checking mechanism based on the heartbeat protocol in Hadoop.

We introduce a new variable *PCR_signed* (the attestation PCR value signed by the AIK) in the *DataNodeCommand* array of the DataNode.java and the NameNode.java. The DataNode will periodically exchange the *DataNodeCommand* array with the NameNode via heartbeat protocol. After receiving the attestation data, the *FSNamesystem.java* running on the NameNode verifies the signature and the PCR values' using the Fingerprint pool (which contains all the clients' PCR values). Finally, a response is generated. For example, if the attestation failed, a penalty may be applied (e.g., shut down the corresponding DataNode).

The PCR values are periodically updated through *ReadPCR*() function in *DataNode.java* via a runtime process (i.e., *ExtendPCR.sh*). The shell script extends the PCR value using previous PCR value and the new measurements produced by *measurements.sml*, as shown in Figure 4.7.

4.6 Performance Evaluation

4.6.1 Test-bed Design and Configuration

To evaluate the security guarantees and the runtime overhead of our authentication protocol, we compare three different Hadoop implementations, namely, baseline Hadoop (no security), Kerberos-based Hadoop and TPM-based Hadoop (our protocol). For Kerberos, we use krb5.x86_64 [15]. For TPM, we use Software-based TPM Emulator because we target the public cloud which relies on virtualized environment. Additionally, according to IBM's introduction of software TPM, an application that can be developed using the software TPM will run using a hardware TPM without changes[88]. On the other hand, using hardware-based TPMs is easier and provides better performance, therefore, the performance results obtained here will be better if hardware-based TPMs were used. To incorporate TPM with Hadoop project, we modify the source code of Hadoop using ant within eclipse [82] and use IAIK jTSS (TCG Software Stack for the Java (tm) Platform [81]) and IAIK jTpmTools (jTT) as the communication interface between Hadoop and TPM. The detailed test-bed configuration is listed below:

Hadoop Deployment: We configure Hadoop in a test-bed environment that involves Ubuntu 10.04 LTE operating system, 2GB memory, 60 GB hard disk, Java version = Oracle Java-1.7.0_67. There are two DataNodes, two TaskTrackers, one NameNode and one JobTracker in the Hadoop system.

Hadoop Deployment with Kerberos: For Hadoop security design with Kerberos, we choose krb5-server.x86_64, krb5-workstation.x86_64, and krb5-devel.x86_64.

Hadoop Deployment with TPM: Hadoop deployment here involves two parts: (1) virtual TPM configuration and jTSS communication interface configuration; (2)Hadoop source code modification environment setup. We use software-based ETH Zurich virtual TPM [109]. The virtual TPM provides all the functionalities of the hardware TPM. However, the virtual TPM is slower than the hardware TPM and, hence, the overhead results presented for our protocol are upper bounds. The runtime overhead of our protocol is expected to be lower when hardware TPM is used.

4.6.2 Runtime Overhead Analysis

It is important to emphasize that developing a new secure authentication protocol, even though is important, is not enough unless it is feasible and practical. Therefore, we have to keep the performance penalty and cost of the added security features within acceptable bounds. In this section, we thoroughly analyze the runtime overhead of our protocol and compare it with the baseline and the Kerberos-based authentication protocols.

The cost of cryptographic processes that prevent replay attacks and ensure data integrity and confidentiality over various communication paths is the same for both TPM-based protocol and Kerberos-based protocol. On the other hand, each protocol has a different overhead that does not exist in the other. The TPM-based protocol incurs a one-time TPM setup overhead that takes place when the TPM in each entity of Hadoop generates the binding keys, AIK, in addition to obtaining certificates for these keys. This is a lightweight overhead and does not create big impact on the

Table 4.2 One Time Overhead of the Proposed System Design

binding key creation	**binding key loading**	**AIK creation**	**AIK loading**	**binding key certification**	**Sum**
˜355.8ms	˜27.1ms	˜108.4ms	˜24.1ms	˜17.0ms	˜532.4ms

day-to-day operations of Hadoop as it is a pre-configuration one time overhead. Table 4.2 shows the pre-configuration runtimes for TPM-based Hadoop under our system configurations. On the other hand, Kerberos-based Hadoop has pre-configuration overhead to perform KDC registration (e.g., adding principal to the KDC database) and to acquire and renew TGT (Ticket Granting Ticket).

The runtime overhead of TPM-based Hadoop is mainly due to: (1) the seal operation to encrypt the $Session\_Key$, (2) the unseal operation to retrieve the $Session\_Key$ whenever required, and (3) the extend and read PCR operation of the heartbeat protocol. The number of seal/unseal operations depends on the life cycle of the $Unsealed\_Session\_Key$ (identical to $Session\_Key$ which is deleted right after the seal operation). The life cycle of the $Unsealed\_Session\_Key$ depends on the verification interval value of the Fingerprint process.

The overall runtime overhead of TPM-based Hadoop (T) can modelled as:

$T = function(Ns \times Ts, Nu \times Tu, Np \times Tp, To)$

Here, Ns is the number of seal operations; Ts is the time cost for one seal operation; Nu is the number of unseal operations; Tu is the time cost for one unseal operation; Np is the number of extend and read PCR operations; Tp is the time cost for one extend and read PCR operation; To is the other extra time costs (e.g., verification of PCR signature or exchange of two random numbers, which incurs negligible time cost compared to the above TPM operations).

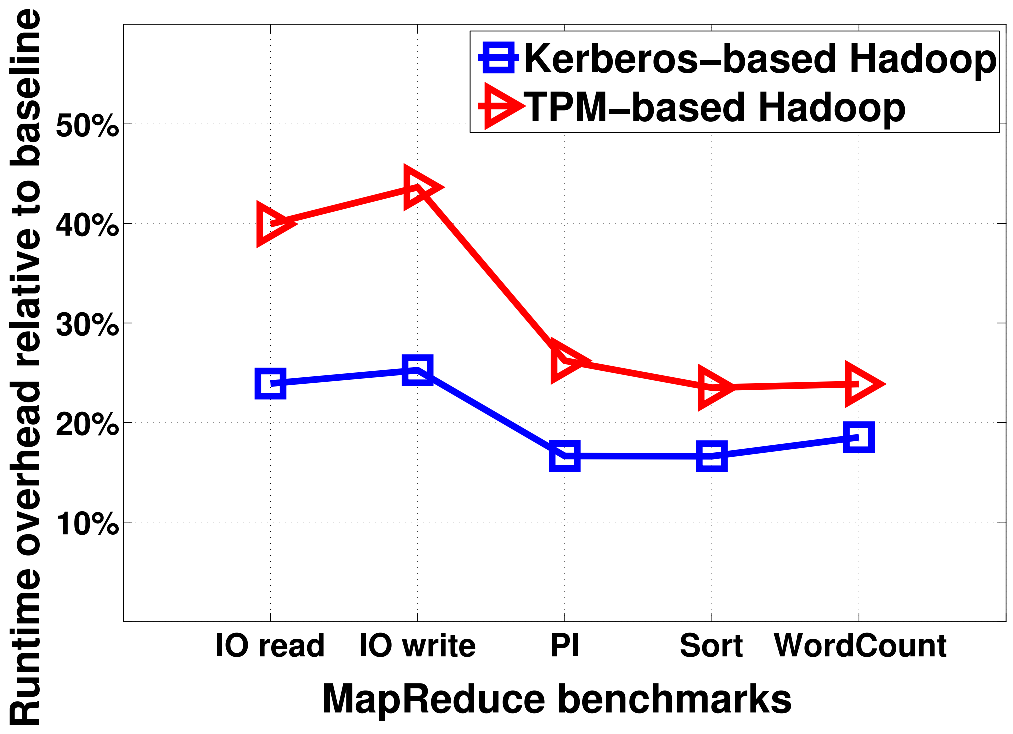


Figure 4.8 Runtime overhead relative to the baseline Hadoop on 5 different types of MapReduce applications.

Overall Runtime Overhead for Different MapReduce Applications: In this experiment, we measure the overall runtime of five different MapReduce applications to compare the overhead of the two secure Hadoop implementations relative to the baseline (none secure) implementation of Hadoop. The first two applications are HDFS write and HDFS read, both of which belong to Hadoop TestDFSIO benchmark. These two test cases focus on testing Hadoop IO performance. The third test case is PI example, which calculates the value of π in a distributed way. It has moderate computation workload and low communication traffic. Another test case is TeraSort benchmark, a sorting application that collectively tests the HDFS and MapReduce layers. The last test case is WordCount example, which has moderate communication traffic and small computation load.

Figure 4.8 presents the results of this experiment. For Kerberos-based Hadoop, the runtime overhead is about 20% relative to the baseline Hadoop. For the TPM-

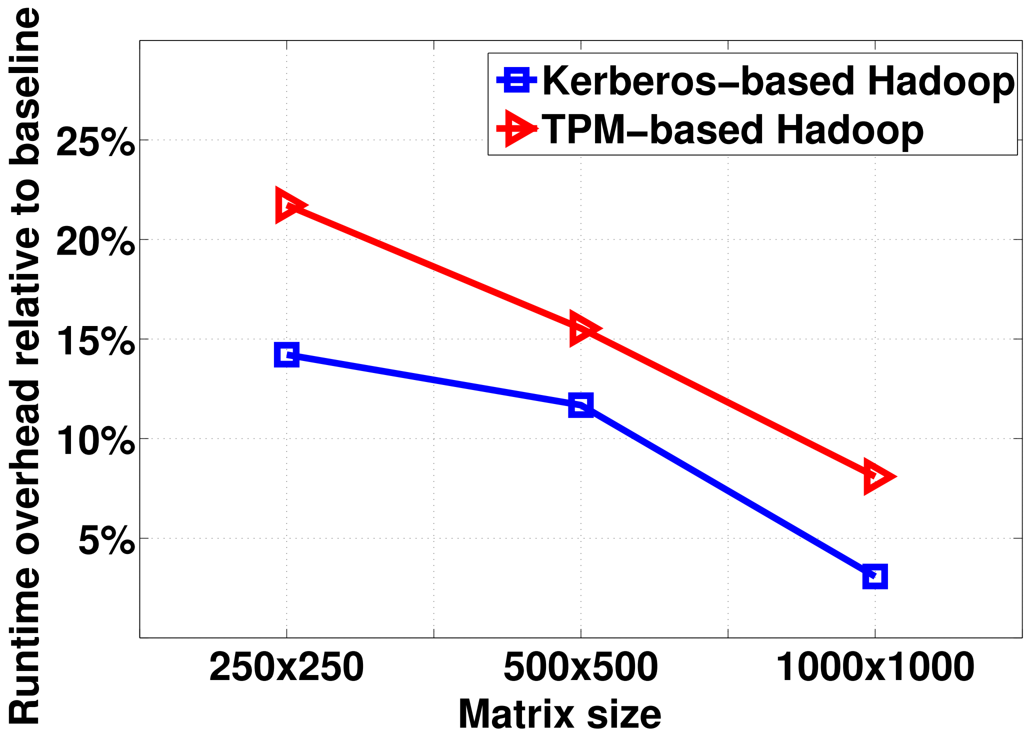


Figure 4.9 Runtime overhead relative to the baseline Hadoop under different workloads.

based Hadoop, the runtime overhead is about 42% for the HDFS write/read and about 25% for the other 3 applications. The relatively high overhead in HDFS write/read is due to the heavy load created by these applications on DataNodes.

Runtime Overhead under Different Workloads: In this experiment, we use the matrix multiplication MapReduce job produced by John Norstad, Northwestern University [60] to test the impact of varying workloads on the overall runtime overhead of the three Hadoop implementations under investigation. We vary the matrix size to generate different workloads. We select three dense matrices with different sizes (i.e., 250x250, 500x500 and 1000x1000). We run the experiment 10 times for each matrix size and compute the average runtime over all the runs. Figure 4.9 shows the results. The figure shows that the TPM-based Hadoop has an extra overhead of about 6.1% over that of Kerberos-base Hadoop due to the TPM operations. The runtime overhead decreases as the workload increases for both Kerberos and TPM-

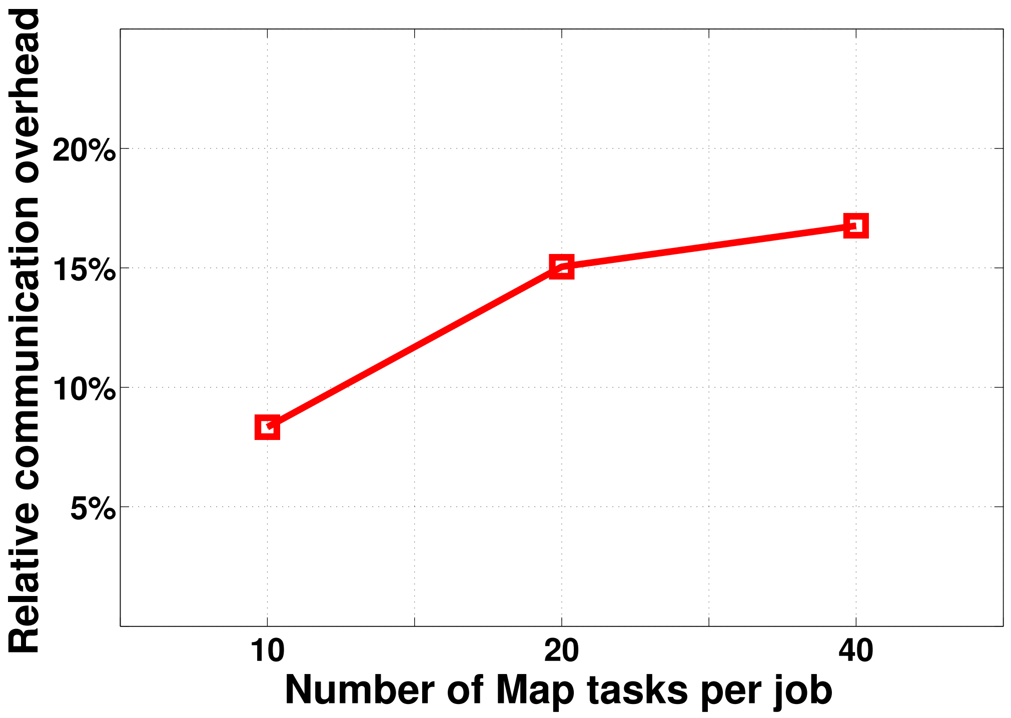


Figure 4.10 Estimated communication overhead of TPM-based Hadoop relative to Kerberos-based Hadoop.

based Hadoop. This trend in the overhead is mainly because of the quadruple increase of the original computational time which makes the approximately linear TPM and Kerberos operations relatively smaller.

Communication Overhead: To test the communication overhead, we choose the Sleep example in Hadoop. In the Sleep example, all tasks do nothing but wait for an assigned amount of time. The purpose of this test scenario is to eliminate the computational overhead by making it the same among all implementations. According to previous non-HDFS applications' experiments, the TPM-based Hadoop has an average of 6.7% extra overhead than Kerberos-based Hadoop. After eliminating this difference, the estimated communication overhead (TPM-based Hadoop compared to the Kerberos-based Hadoop) is summarized in Figure 4.10. The communication overhead increases as the number of Map tasks increases. This is due to the increase in the number of TPM operations for the additional RPC sessions that deliver Map

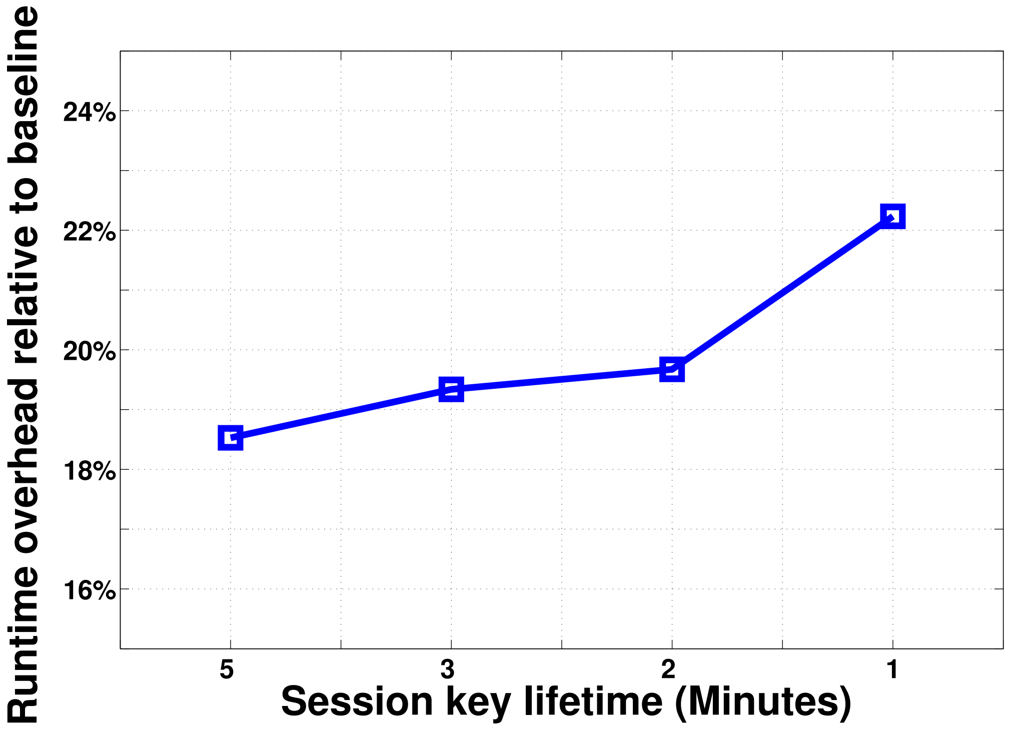


Figure 4.11 Runtime overhead of TPM-based Hadoop relative to baseline with variable session key lifetime.

tasks. TPM has an average of 13.4% communication overhead over Kerberos. This extra communication overhead includes the exchange of the random numbers for session key establishment and the Fingerprint verification operations.

Variable Session Key Lifetime and Heartbeat Interval Values: As mentioned in Section 4.3.4, the number of NameNode side TPM operations decreases significantly as we replace the TPM seal and unseal operations with the Fingerprint verification that is carried out outside the TPM.

Figure 4.11 presents the runtime overhead for different session key lifetimes without the Fingerprint verification mechanism. Shorter session key lifetime achieves better security guarantees at the cost of higher runtime overhead. The overhead is mainly due to the unseal operations to retrieve the session key. On the other hand, the Fingerprint verification helps to offload the NameNode's TPM while maintaining the same security guarantees by carefully adjusting the attestation interval value which is

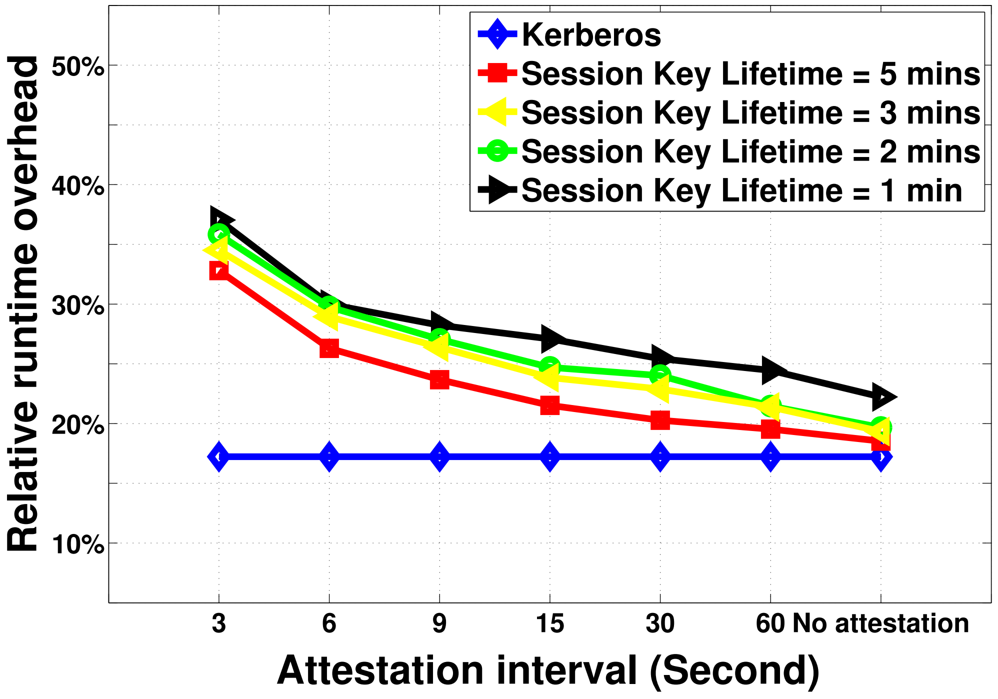


Figure 4.12 Runtime overhead of TPM-based Hadoop relative to the baseline with variable Fingerprint attestation intervals for various session key lifetimes.

based on the heartbeat rate. Figure 4.12 shows the runtime overhead relative to the baseline Hadoop for the PI example for various attestation intervals. Intuitively, the figure shows that the higher the attestation interval, the lower the overhead. Also, the higher the lifetime of the session key, the lower the overhead. The former trend is due to the lower number of Fingerprint operations with high attestation intervals; while the latter trend is due to lower number of seal/unseal operations with high session key lifetime. Therefore, by turning the session key lifetime and the attestation interval, we can control the tradeoff between the overhead and security in TPM-based Hadoop.

4.7 Related Work

In early 2013, Project Rhino was launched by Intel as an open source project with a goal to improve the security capabilities of Hadoop. The group proposes Task HADOOP-9392 (Token-Based Authentication and Single Sign-On) which intends to support tokens for many authentication mechanisms such as Lightweight Directory Access Protocol (LDAP), Kerberos, X.509 Certificate authentication, SQL authentication, and Security Assertion Markup Language (SAML) [10]. The project mainly focuses on how to extend the current authentication framework to a standard interface for supporting different types of authentication protocols. Nevertheless, all these authentication protocols, including Kerberos, are software-based methods that are vulnerable to privileged user manipulations. An insider or possibly an outsider could indirectly collect users' credentials through, for example, the installation of malware/spyware tools on the machines they have access to in a way that is transparent to the victims. Rhino design trades off flexibility with complexity. Overall, it enhances the flexibility of the authentication mechanisms at the cost of increasing the complexity of the system.

In [7], the author proposes a TPM-based Kerberos protocol. The proposed protocol is able to issue tickets bound to the client platform through integrating PCA functionality into the Kerberos authentication server (AS) and remote attestation functionality into the Ticket-Granting Server (TGS). However, the proposed mechanism does not provide any attestation for Hadoop's internal components. Nothing can prevent malicious Hadoop insiders from tampering with internal Hadoop components. In this work, we use TPM functionalities to perform authentication directly inside Hadoop and eliminate the need for any trusted-third-party.

In [35], the authors propose a Trusted MapReduce (TMR) framework that integrates MapReduce systems with the TCG (i.e., Trusted Computing Group) trusted computing infrastructure. They present an attestation protocol between the JobTracker and the TaskTracker to ensure the integrity of each party in the MapReduce framework. However, they mainly focus on the integrity verification of the Hadoop MapReduce framework without addressing the authentication issues of Hadoop's HDFS and Clients. Therefore, the authors do not provide a general authentication framework for the whole Hadoop ecosystem.

In [8], the authors present a design of a trusted cloud computing platform (TCCP) based on TPM technologies. The proposed design guarantees confidential execution of guest VMs, and allows users to attest to the IaaS provider to determine if the service is secure before they launch their VMs. Nevertheless, they do not provide details about how their design will be implemented and no performance evaluations are provided. Also, they fail to provide a complete authentication framework among all the components of Hadoop.

4.8 Conclusion

In this work, we design and implement a TPM-based authentication protocol for Hadoop that provides strong mutual authentication between any internally interacting Hadoop entities, in addition to mutually authenticating with external clients. The bind and seal operations supported by the TPM protect against malicious insiders since insiders cannot change the machine state without affecting the PCR values. Furthermore, our protocol alleviates the use of the trusted third party by using the AIK certification. Moreover, we compare the security features and overhead of our protocol with the state-of-the-art protocols and show that our protocol provides better security guarantees with acceptable overhead.

In the future work, we will tighten the security requirements of the NameNode by removing the assumption of partial trust.

CHAPTER 5

SUMMARY AND FUTURE DIRECTIONS

5.1 Summary

In this dissertation, we present three types of entity authentication mechanisms: (1) End user authentication: a novel and robust authentication factor based on network communications latency (Chapter 2); (2) Web server authentication: a novel and robust phishing detection feature (Chapter 3); and (3) Platform and service authentication: robust insider attacks countermeasure for Hadoop: design & implementation (Chapter 4);

In the first work, we propose a new authentication factor based on Network Round Trip Time ($NRTT$) and show how $NRTT$ can be used to uniquely and securely identify login locations and hence can support location-based web authentication mechanisms. Two research challenges are identified and resolved. The first research challenge is that the naive measurement ans usage of $NRTT$ allow the attackers to easily manipulate or brute force the $NRTT$ profile of the user. We address this research challenging by introducing a novel network segment: Delay Mask (DM), to hide the $NRTT$ from the attackers (as well as the legitimate users). In addition, with multiple DMs deployment (i.e., multiple $NRTT$ profiles), the sample space of $NRTT$ is enlarged such that manipulation of the $NRTT$ profiles is more difficulty. The second research challenge is the network instabilities. First, we classify these network instabilities into three different categories: (1) **Instantaneous instabilities** are instabilities which lead to transient changes in communications latency; (2) **Long-term instabilities** are instabilities that stay long enough to affect all or most of the of profiling signals; and (3)**Routing instabilities** are instabilities that result in permanent changes in network communications latency. We address these network instabilities through three novel algorithms, namely, outlier removal based on Gaussian approximation, shared increment removal algorithm and long term dynamic temporal profiling. We conduct extensive experiments to evaluate Security-Usability-Deployability properties of NRTT as an authentication factor and compare it with state-of-the-art authentication mechanisms.

In the second work, we introduce a new robust phishing phishing detection feature based on Network Round Trip Time, $NRTT$. In addition, we design a new phishing detection framework by integrating a set of selected phishing detection features together a phishing target identification algorithm. The experimental results shows the usage of $NRTT$ could increase the detection accuracy while maintaining a low false positive rate. The comparative evaluation of the proposed scheme shows that it has comparable performance results compared to the state-of-the-art schemes.

In the third work, we design and implement a TPM-based authentication protocol for Hadoop that provides strong mutual authentication between any internally interacting Hadoop entities. The bind and seal operations supported by the TPM protect against malicious insiders since insiders cannot change the machine state without affecting the PCR values. Furthermore, our protocol alleviates the use of the trusted third party by using the AIK certification. Moreover, we compare the security features and overhead of our protocol with the state-of-the-art protocols and show that our protocol provides better security guarantees with acceptable overhead.

5.2 Future Direction

In the future, we plan to further investigate the effectiveness and security impacts of using $NRTT$ and other phishing detection features in a more comprehensive way.

Firstly, we plan to investigate and evaluate more phishing detection features based on both the detection accuracy and the security impacts (e.g., unforgerybility) because it is paramount for phishing detection approaches to carefully select the features that strike the right balance between detection accuracy and robustness in the face of potential manipulations.

In addition, we plan to leverage Deep Learning (DL) algorithms to improve the performance of phishing detection schemes because DL could be a viable alternative to traditional machine learning algorithms (e.g., SVM, LR), especially when handling complex and large scale datasets.

BIBLIOGRAPHY

[1] 2016 Phishing Trends & Intelligence Report: Hacking the Human. https://info.phishlabs.com/pti-report-download. [Online; last accessed on 8-April-2018].

[2] Anti-phishing extension: Netcraft. http://toolbar.netcraft.com/. [Online; last accessed on 8-April-2018].

[3] Anti-Phishing Working Group. http:/www.antiphishing.org/. [Online; last accessed on 8-April-2018].

[4] ConsumerReports: Smartphone thefts. www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm. [Online; last accessed on 8-April-2018].

[5] The Cyberwire: Breaking news from RSA. http://thecyberwire.com/issues/issues2015/April/CyberWire_2015_04_21.html. [Online; last accessed on 8-April-2018].

[6] Dell secure work: BGP Hijacking for Cryptocurrency Profit. http://www.secureworks.com/cyber-threat-intelligence/threats/bgp-hijacking-for-cryptocurrency-profit/. [Online; last accessed on 8-April-2018].

[7] Facebook Connect. https://developers.facebook.com/blog/post/2008/05/09/announcing-facebook-connect/. [Online; last accessed on 8-April-2018].

[8] Gmail: Detecting suspicious account activity. http://googleonlinesecurity.blogspot.com/2010/03/detecting-suspicious-account-activity.html. [Online; last accessed on 8-April-2018].

[9] Google Safe Browsing. https://developers.google.com/safe-browsing/. [Online; last accessed on 8-April-2018].

[10] LastPass. https://lastpass.com/. [Online; last accessed on 8-April-2018].

[11] NIST/SEMATECH, e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/.. [Online; last accessed on 8-April-2018].

[12] Phishing Safety: Is HTTPS Enough? http://blog.trendmicro.com/trendlabs-security-intelligence/phishing-safety-is-https-enough/. [Online; last accessed on 8-April-2018].

[13] RSA SecurID. http://www.emc.com/security/rsa-securid/index.htm. [Online; last accessed on 8-April-2018].

[14] RSA SecurID. `http://www.emc.com/collateral/solution-overview/10695-sidtfa-sb.pdf`. [Online; last accessed on 8-April-2018].

[15] Stat 300 materials 7-3a. `http://flc.losrios.edu/~eitel/Stat\%20300/S-300\%20Main\%20Web\%20Page.htm`. [Online; last accessed on 8-April-2018].

[16] Symantec: Internet Security Threat Report. `https://www.symantec.com/content/en/us/enterprise/other_resources/21347933_GA_RPT-internet-security-threat-report-volume-20-2015.pdf`. [Online; last accessed on 8-April-2018].

[17] Yahoo On-Demand Password. `https://www.yahoo.com/tech/yahoo-introduces-on-demand-passwords-uses-your-113794671449.html`. [Online; last accessed on 8-April-2018].

[18] Greg Aaron and Rod Rasmussen. Global phishing survey: trends and domain name use in 2h2009. *Anti-Phishing Working Group (APWG), Lexington, MA*, 2010.

[19] Saeed Abu-Nimeh and Suku Nair. Bypassing security toolbars and phishing filters via dns poisoning. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, 2008.

[20] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *the anti-phishing working groups 2nd annual eCrime researchers summit*. ACM, 2007.

[21] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *the Special Interest Group on Management of Data (SIGMOD)*. ACM, 2004.

[22] Fadi A Aloul, Syed Zahidi, and Wassim El-Hajj. Two factor authentication using mobile phones. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pages 641–644, 2009.

[23] Anti-Phishing Working Group et al. APWG Phishing trends reports 2010-2016. `https://www.antiphishing.org/resources/apwg-reports/`. [Online; last accessed on 8-April-2018].

[24] Emma Ban. How important are false positives in measuring the quality of an antimalware engine? `http://oemhub.bitdefender.com/importance-of-false-positives-for-antimalware-engine-quality`. [Online; last accessed on 8-April-2018].

[25] Karyn Benson, Rafael Dowsley, and Hovav Shacham. Do you know where your cloud files are? In *the 3rd ACM workshop on Cloud computing security workshop*, pages 73–82, 2011.

[26] André Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paaß, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of computer security*, 18(1):7–35, 2010.

[27] Mark Berman, Jeffrey S Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. GENI: A federated testbed for innovative network experiments. *Computer Networks*, 61:5–23, 2014.

[28] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *the 3rd ACM workshop on Artificial intelligence and security*, 2010.

[29] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy (SP)*, pages 553–567, 2012.

[30] John Brainard, Ari Juels, Ronald L Rivest, Michael Szydlo, and Moti Yung. Fourth-factor authentication: somebody you know. In *13th ACM conference on Computer and communications security*, 2006.

[31] Deanna D Caputo, Shari Lawrence Pfleeger, Jesse D Freeman, and M Eric Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38, 2014.

[32] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[33] Shehzad Ashraf Chaudhry, Mohammad Sabzinejad Farash, Husnain Naqvi, Saru Kumari, and Muhammad Khurram Khan. An enhanced privacy preserving remote user authentication scheme with provable security. *Security and Communication Networks*, 2015.

[34] Teh-Chung Chen, Torin Stepan, Scott Dick, and James Miller. An anti-phishing system employing diffused information. *ACM Transactions on Information and System Security (TISSEC)*, 16(4):16, 2014.

[35] Sonia Chiasson, Elizabeth Stobert, Alain Forget, Robert Biddle, and Paul C Van Oorschot. Persuasive cued click-points: Design, implementation, and evaluation of a knowledge-based authentication mechanism. *IEEE Transactions on Dependable and Secure Computing*, 2012.

[36] Neil Chou, Robert Ledesma, Yuka Teraguchi, John C Mitchell, et al. Client-side defense against web-based identity theft. In *the Network and Distributed System Security Symposium*, 2004.

[37] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM Special Interest Group on Data Communications (SIGCOM) Computer Communication Review*, 2003.

[38] Giovanni Comarela, Gonca Gürsun, and Mark Crovella. Studying interdomain routing over long timescales. In *the conference on Internet measurement conference*, pages 227–234. ACM, 2013.

[39] Vedat Coskun, Kerem Ok, and Busra Ozdenizci. *Near Field Communication (NFC): From Theory to Practice.* John Wiley & Sons, Hoboken, NJ, 2011.

[40] Dorothy E Denning and Peter F MacDoran. Location-based authentication: Grounding cyberspace for better security. *Computer Fraud & Security*, 1996(2):12–16, 1996.

[41] Zuochao Dou, Issa Khalil, and Abdallah Khreishah. CLAS: A novel communications latency based authentication scheme. *Security and Communication Networks*, 2017.

[42] Zuochao Dou, Issa Khalil, and Abdallah Khreishah. A novel and robust authentication factor based on network communications latency. *IEEE Systems Journal*, 2017.

[43] Zuochao Dou, Issa Khalil, Abdallah Khreishah, and Ala Al-Fuqaha. Robust insider attacks countermeasure for hadoop: Design and implementation. *IEEE Systems Journal*, 2017.

[44] Chuck Fraleigh, Sue Moon, Bryan Lyles, Chase Cotton, Mujahid Khan, Deb Moll, Rob Rockell, Ted Seely, and Sprint Christophe Diot. Packet-level traffic measurements from the sprint ip backbone. *IEEE Network*, 2003.

[45] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F Gryniewicz, and Yixin Jin. An architecture for a global internet host distance estimation service. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 210–217, 1999.

[46] Simson L Garfinkel. Email-based identification and authentication: An alternative to PKI?, 2003.

[47] Sophie Gastellier-Prevost, Gustavo Gonzalez Granadillo, and Maryline Laurent. A dual approach to detect pharming attacks at the client-side. In *4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2011.

[48] Artyom Gavrichenkov. Breaking HTTPS with BGP hijacking. *Black Hat Briefings*, 2015.

[49] Ammar Gharaibeh, Mohammad A Salahuddin, Sayed J Hussini, Abdallah Khreishah, Issa Khalil, Mohsen Guizani, and Ala Al-Fuqaha. Smart cities: A survey on data management, security and enabling technologies. *IEEE Communications Surveys & Tutorials*, 2017.

[50] Eric Grosse and Mayank Upadhyay. Authentication at scale. *IEEE Security & Privacy*, 11(1):15–22, 2013.

[51] Anti-Phishing Working Group et al. Phishing activity trends report. 2015. `https://www.antiphishing.org/resources/apwg-reports/`. [Online; last accessed on 8-April-2018].

[52] Ismail Hababeh, Issa Khalil, and Abdallah Khreishah. Designing high performance web-based computing services to promote telemedicine database management system. *IEEE Transactions on Services Computing*, 8(1):47–64, 2015.

[53] Steffen Hallsteinsen, Ivar Jorstad, et al. Using the mobile phone as a security token for unified authentication. In *the Second International Conference on Systems and Networks Communications (ICSNC)*, pages 68–68. IEEE, 2007.

[54] Aaron L-F Han, Derek F Wong, and Lidia S Chao. Password cracking and countermeasures in computer security: A survey. *arXiv preprint arXiv:1411.7803*, 2014.

[55] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1402–1413, 2016.

[56] Health Information Trust Alliance (HITRUST). U.S. Healthcare Data Breach Trends. `https://hitrustalliance.net/breach-reports/`. [Online; last accessed on 8-April-2018].

[57] Scott Hogg. Address authenticaion. *The Internet Protocol Journal*, 2013.

[58] Jason Hong. The state of phishing attacks. *Communications of the ACM*, 55(1):74–81, 2012.

[59] Russell Housley, W Polk, Warwick Ford, and David Solo. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile, 2002.

[60] Xin Hu and Zhuoqing Morley Mao. Accurate real-time identification of IP prefix hijacking. In *IEEE Security & Privacy*, 2007.

[61] Zuochao Dou Issa Khalil and Abadallah Khreishah. Your credentials are compromised, do not panic: You can be well protected. *ACM Asia Conference on Computer & Communications Security*, 2016.

[62] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 2007.

[63] Ari Juels and Ronald L Rivest. Honeywords: Making password-cracking detectable. In *the 2013 ACM SIGSAC conference on Computer & communications security*, 2013.

[64] Poul-Henning Kamp, P Godefroid, M Levin, D Molnar, P McKenzie, R Stapleton-Gray, B Woodcock, and G Neville-Neil. Linkedin password leak: Salt their hide. *ACM Queue*, 10(6):20, 2012.

[65] Chris Karlof, Umesh Shankar, J Doug Tygar, and David Wagner. Dynamic pharming attacks and locked same-origin policies for web browsers. In *the 14th ACM conference on computer and communications security*, pages 58–71, 2007.

[66] Natallia Katenka, Elizaveta Levina, and George Michailidis. Local vote decision fusion for target detection in wireless sensor networks. *IEEE Transactions on Signal Processing*, 56(1):329–338, 2008.

[67] Ethan Katz-Bassett, John P John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards ip geolocation using delay and topology measurements. In *the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84, 2006.

[68] Issa Khalil and Saurabh Bagchi. Secos: key management for scalable and energy efficient crypto on sensors. *In IEEE Dependable Systems and Networks (DSN)*, 2003.

[69] Issa Khalil, Saurabh Bagchi, and Ness Shroff. Analysis and evaluation of secos, a protocol for energy efficient and secure communication in sensor networks. *Ad Hoc Networks*, 5(3):360–391, 2007.

[70] Issa Khalil, Zuochao Dou, and Abdallah Khreishah. Tpm-based authentication mechanism for apache hadoop. In *International Conference on Security and Privacy in Communication Systems*, pages 105–122. Springer, 2014.

[71] Issa Khalil, Zuochao Dou, and Abdallah Khreishah. Your credentials are compromised, do not panic: You can be well protected. In *the 11th ACM Asia Conference on Computer & Communications Security*, 2016.

[72] Issa Khalil, Ismail Hababeh, and Abdallah Khreishah. Secure inter cloud data migration. In *the 7th International Conference on Information and Communication Systems (ICICS)*, pages 62–67. IEEE, 2016.

[73] Issa Khalil, Ting Yu, and Bei Guan. Discovering malicious domains through passive dns data graph analysis. In *the 11th ACM on Asia Conference on Computer and Communications Security*, pages 663–674, 2016.

[74] Issa M Khalil, Abdallah Khreishah, and Muhammad Azeem. Cloud computing security: a survey. *Computers*, 3(1):1–35, 2014.

[75] Majid Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[76] Lawrence A Klein. A boolean algebra approach to multiple sensor voting fusion. *IEEE transactions on Aerospace and Electronic systems*, 29(2):317–327, 1993.

[77] Nicholas Kuechler. GeoPing: Ping test from multiple geographic locations. `http://www.geopinginfo.com/`. [Online; last accessed on 8-April-2018].

[78] Ponnurangam Kumaraguru, Yong Rhee, Steve Sheng, Sharique Hasan, Alessandro Acquisti, Lorrie Faith Cranor, and Jason Hong. Getting users to pay attention to anti-phishing education: evaluation of retention and transfer. In *the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 70–81. ACM, 2007.

[79] Ponnurangam Kumaraguru, Steve Sheng, Alessandro Acquisti, Lorrie Faith Cranor, and Jason Hong. Teaching johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT)*, 10(2):7, 2010.

[80] Minseok Kwon, Zuochao Dou, Wendi Heinzelman, Tolga Soyata, He Ba, and Jiye Shi. Use of network latency profiling and redundancy for cloud server selection. In *IEEE 7th International Conference on Cloud Computing, 2014*.

[81] Craig Labovitz, G Robert Malan, and Farnam Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 1998.

[82] Mohit Lad, Jong Han Park, Tiziana Refice, and Lixia Zhang. A study of internet routing stability using link weight. Technical report, UCLA/CSD-080003, 2008.

[83] Stevens Le Blond, Adina Uritesc, Cédric Gilbert, Zheng Leong Chua, Prateek Saxena, and Engin Kirda. A look at targeted attacks through the lense of an NGO. In *USENIX Security Symposium*, pages 543–558, 2014.

[84] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[85] Gastón L'Huillier, Alejandro Hevia, Richard Weber, and Sebastían Ríos. Latent semantic analysis and keyword extraction for phishing classification. In *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*, pages 129–131. IEEE, 2010.

[86] Hongwei Li, Yuanshun Dai, Ling Tian, and Haomiao Yang. Identity-based authentication for cloud computing. In *IEEE International Conference on Cloud Computing*, pages 157–166. Springer, 2009.

[87] Gang Liu, Bite Qiu, and Liu Wenyin. Automatic detection of phishing target from phishing webpage. In *the 20th International Conference on Pattern Recognition (ICPR)*, pages 4153–4156. IEEE, 2010.

[88] Chen Change Loy, Weng Kin Lai, and Chee Peng Lim. Keystroke patterns classification using the artmap-fd neural network. In *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, volume 1, pages 61–64. IEEE, 2007.

[89] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *the 15th ACM international conference on knowledge discovery and data mining (SIGKDD)*, pages 1245–1254, 2009.

[90] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. *arXiv preprint arXiv:1510.06501*, 2015.

[91] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *the 4th international conference on Security and privacy in communication netowrks*. ACM, 2008.

[92] James W Mickens, John R Douceur, William J Bolosky, and Brian D Noble. StrobeLight: Lightweight availability mapping and anomaly detection. In *USENIX Annual Technical Conference*, 2009.

[93] Marino Miculan and Caterina Urban. Formal analysis of Facebook Connect single sign-on authentication protocol. In *the 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 11, pages 22–28. Citeseer, 2011.

[94] Ioan-Cosmin Mihai and Laurentiu Giurea. Management of elearning platforms security. In *The International Scientific Conference eLearning and Software for Education*, volume 1, page 422. "Carol I" National Defence University, 2016.

[95] Fabian Monrose and Aviel D Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation computer systems*, 16(4):351–359, 2000.

[96] Thomas Morris. Trusted platform module. In *Encyclopedia of Cryptography and Security*. Springer, 2011.

[97] James A Muir and Paul C Van Oorschot. Internet geolocation: Evasion and counterevasion. *ACM computing surveys*, 42(1):4, 2009.

[98] Paul Mutton. Fraudsters seek to make phishing sites undetectable by content filters, 2011. `https://news.netcraft.com/archives/2005/05/12/fraudsters_seek_to_make_phishing_sites_undetectable_by_content_filters.html`. [Online; last accessed on 8-April-2018].

[99] Clifford Neuman and Theodore Ts' O. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

[100] LLC OpenDNS. Phishtank: an anti-phishing site. `https://www.phishtank.com/`. [Online; last accessed on 8-April-2018].

[101] Venkata N Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *ACM SIGCOMM Computer Communication Review*, 2001.

[102] Rajesh Krishna Panta, Saurabh Bagchi, and Issa M Khalil. Efficient wireless reprogramming through reduced bandwidth usage and opportunistic sleeping. *Ad Hoc Networks*, 7(1):42–62, 2009.

[103] Bimal Parmar. Protecting against spear-phishing. *Computer Fraud & Security*, 2012(1):8–11, 2012.

[104] Andreas Pashalidis and Chris J Mitchell. Impostor: A single sign-on system for use from untrusted devices. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2004.

[105] Maja Pusara and Carla E Brodley. User re-authentication via mouse movements. In *the ACM workshop on visualization and data mining for computer security*, pages 1–8, 2004.

[106] Gowtham Ramesh, Ilango Krishnamurthi, and K Sampath Sree Kumar. An efficacious method for detecting phishing webpages through target domain identification. *Decision Support Systems*, 61:12–22, 2014.

[107] David Recordon and Drummond Reed. OpenID 2.0: a platform for user-centric identity management. In *the second ACM workshop on Digital identity management*, 2006.

[108] Robert W Reeder and Stuart Schechter. When the password doesn't work: Secondary authentication for websites. *IEEE Security & Privacy*, 2011.

[109] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. BGP routing stability of popular destinations. In *the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002.

[110] Douglas Reynolds. Gaussian mixture models* MIT lincoln laboratory, 244 wood st. *Lexington, MA.*

[111] Martin S Ridout. An improved threshold approximation for local vote decision fusion. *IEEE Transactions on Signal Processing*, 2013.

[112] Arun Ross, Jidnya Shah, and Anil K Jain. From template to image: Reconstructing fingerprints from minutiae points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[113] Ulrich Rührmair, Jan Sölter, and Frank Sehnke. On the foundations of physical unclonable functions. *IACR Cryptology ePrint Archive*, 2009:277, 2009.

[114] Aman Shaikh, Anujan Varma, Lampros Kalampoukas, and Rohit Dube. Routing stability in congested networks: Experimentation and analysis. In *ACM SIGCOMM Computer Communication Review*, 2000.

[115] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382. ACM, 2010.

[116] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. In *the 3rd symposium on Usable privacy and security*, pages 88–99. ACM, 2007.

[117] Evan R Sparks and Evan R Sparks. A security assessment of trusted platform modules computer science technical report. Technical report, 2007.

[118] Spear Phishing: Real Life Examples. `http://resources.infosecinstitute.com/spear-phishing-real-life-examples/`. [Online; last accessed on 8-April-2018].

[119] Suresh Sriniwas. An introduction to HDFS federation. `https://hortonworks.com/blog/an-introduction-to-hdfs-federation/`. [Online; last accessed on 8-April-2018].

[120] Mitsuho Tahara, Naoki Tateishi, Toshio Oimatsu, and Souhei Majima. A method to detect prefix hijacking by using ping tests. In *Challenges for Next Generation Network Operations and Service Management*. Springer, 2008.

[121] Hai Tao. *Pass-Go, a new graphical password scheme*. PhD thesis, University of Ottawa, Canada, 2006.

[122] CA Technology. Advanced Authentication Methods: Software vs. Hardware. `http://www3.ca.com/~/media/Files/whitepapers/ebook-advanced-authenticaiton-methods.PDF`. [Online; last accessed on 8-April-2018].

[123] Pin Shen Teh, Andrew Beng Jin Teoh, Thian Song Ong, and Han Foon Neo. Statistical fusion approach on keystroke dynamics. In *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, 2007.

[124] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *2011 IEEE Symposium on Security and Privacy*, pages 447–462, 2011.

[125] Visa Antero Vallivaara, Mirko Sailio, and Kimmo Halunen. Detecting man-in-the-middle attacks on non-mobile systems. In *ACM Conference on Data and Application Security and Privacy*, 2014.

[126] Pierre-Antoine Vervier, Olivier Thonnard, and Marc Dacier. Mind your blocks: On the stealthiness of malicious BGP hijacks. In *the Network and Distributed System Security Symposium*, 2015.

[127] Pierre-Antoine Vervier, Olivier Thonnard, and Marc Dacier. Mind your blocks: On the stealthiness of malicious bgp hijacks. In *the Network and Distributed System Security Symposium*, 2015.

[128] Jingguo Wang, Tejaswini Herath, Rui Chen, Arun Vishwanath, and H Raghav Rao. Research article phishing susceptibility: An investigation into the processing of a targeted spear phishing email. *IEEE transactions on professional communication*, 55(4):345–362, 2012.

[129] James Wayman, Anil Jain, Davide Maltoni, and Dario Maio. *An introduction to biometric authentication systems*. Springer, 2005.

[130] Daphna Weinshall. Cognitive authentication schemes safe against spyware. In *IEEE Symposium on Security & Privacy, 2006.*

[131] Liu Wenyin, Gang Liu, Bite Qiu, and Xiaojun Quan. Antiphishing through phishing target discovery. *IEEE Internet Computing*, 16(2):52–61, 2012.

[132] Greg White. Trends in Hardware Authentication. Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2015.

[133] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *the Network and Distributed System Security Symposium*, volume 10, 2010.

[134] Alexander Wiesmaier, Marcus Fischer, Marcus Lippert, and Johannes Buchmann. Outflanking and securely using the pin/tan-system. *arXiv preprint cs/0410025*, 2004.

[135] Wikipedia. Avalanche (phishing group). `https://en.wikipedia.org/w/index.php?title=Avalanche_(phishing_group)&oldid=739184980`. [Online; last accessed on 8-April-2018].

[136] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):21, 2011.

[137] John Yearwood, Musa Mammadov, and Arunava Banerjee. Profiling phishing emails based on hyperlink information. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 120–127. IEEE, 2010.

[138] Shruti Nargundkar Yu Weider and Nagapriya Tiruthani. A phishing vulnerability analysis of web based systems. In *IEEE Symposium on Computers and Communications*, pages 326–331, 2008.

[139] Yulong Zhang, Zhaonfeng Chen, Hui Xue, and Tao Wei. Fingerprints on mobile devices: Abusing and eaking. In *Black Hat Conference*, 2015.

[140] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation computer systems*, 28(3):583–592, 2012.